

**Laporan Tugas Besar 3 IF2211 Strategi Algoritma  
Semester II tahun 2024/2025  
Pemanfaatan Pattern Matching untuk Membangun Sistem ATS  
(Applicant Tracking System) Berbasis CV Digital**



**Disusun oleh:**

**Jonathan Kenan Budianto (13523139)**

**Mahesa Fadhillah Andre (13523140)**

**Lukas Raja Agripa (13523158)**

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung**

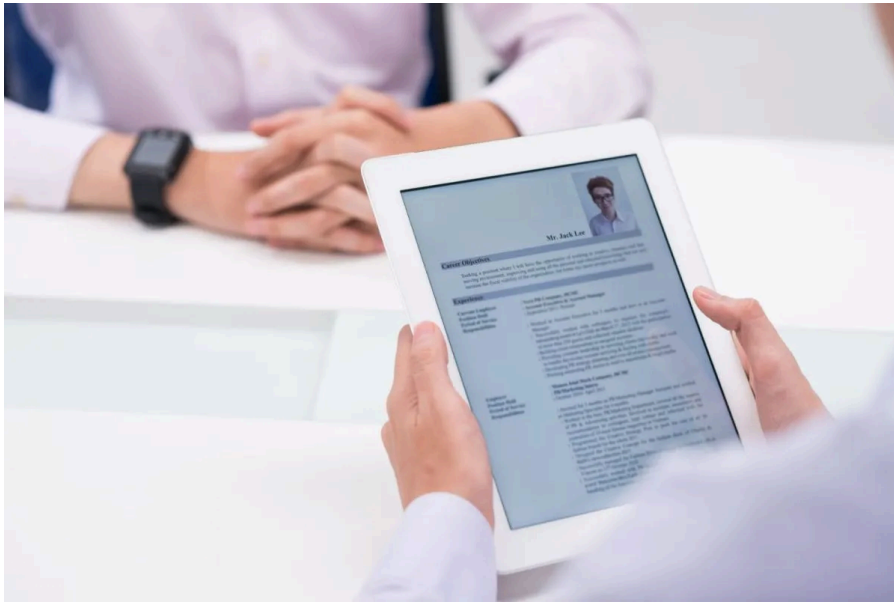
**Jl. Ganesha 10, Bandung 40132**

**2025**

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Bab 1 : Deskripsi Tugas</b>	<b>3</b>
<b>Penjelasan Implementasi</b>	<b>4</b>
<b>Penggunaan Program</b>	<b>7</b>
<b>Bab 2 : Landasan Teori</b>	<b>9</b>
2.1 Algoritma Knuth-Morris-Pratt	9
2.2 Algoritma Boyer-Moore	10
2.3 Web/GUI yang dibangun	11
<b>Bab 3 : Analisis Pemecahan Masalah</b>	<b>13</b>
3.1 Langkah-langkah pemecahan masalah	13
3.2 Proses pemetaan masalah menjadi elemen-elemen algoritma KMP dan BM.	15
3.3 Fitur fungsional dan arsitektur aplikasi web yang dibangun	18
3.4 Contoh ilustrasi kasus	23
<b>Bab 4 : Implementasi dan Pengujian</b>	<b>23</b>
4.1 Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).	24
4.1.1 Struktur Data	24
4.1.2 Moduk dan Fungsi	25
4.1.3 Prosedur	27
4.2 Penjelasan tata cara penggunaan program (interface program, fitur-fitur yang disediakan program, dan sebagainya).	28
4.3 Hasil pengujian minimal 4 pencarian dengan variasi keyword, algoritma, dan panjang keyword yang berbeda-beda.	32
4.4 Analisis hasil pengujian.	32
<b>Bab 5 : Penutup</b>	<b>33</b>
5.1 Kesimpulan	33
5.2 Saran	33
5.3 Refleksi	33
<b>Lampiran</b>	<b>34</b>
<b>Daftar Pustaka</b>	<b>35</b>

# Bab 1 : Deskripsi Tugas



**Gambar 1.** CV ATS dalam Dunia Kerja  
(Sumber: <https://www.antaraneews.com/> )

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan proses rekrutmen tenaga kerja telah mengalami perubahan signifikan dengan memanfaatkan teknologi untuk meningkatkan efisiensi dan akurasi. Salah satu inovasi yang menjadi solusi utama adalah Applicant Tracking System (ATS), yang dirancang untuk mempermudah perusahaan dalam menyaring dan mencocokkan informasi kandidat dari berkas lamaran, khususnya Curriculum Vitae (CV). ATS memungkinkan perusahaan untuk mengelola ribuan dokumen lamaran secara otomatis dan memastikan kandidat yang relevan dapat ditemukan dengan cepat.

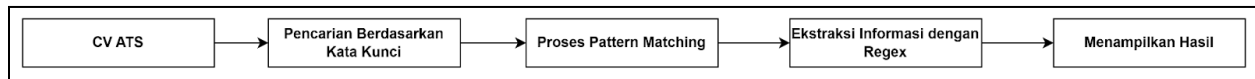
Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan riwayat pendidikan secara efisien. Pattern matching menjadi solusi ideal dalam menghadapi tantangan ini.

Pattern matching adalah teknik untuk menemukan dan mencocokkan pola tertentu dalam teks. Dalam konteks ini, algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) sering digunakan karena keduanya menawarkan efisiensi tinggi untuk pencarian teks di dokumen besar. Algoritma ini memungkinkan sistem ATS untuk mengidentifikasi informasi penting dari CV pelamar dengan kecepatan dan akurasi yang optimal.

Di dalam Tugas Besar 3 ini, Anda diminta untuk mengimplementasikan sistem yang dapat melakukan deteksi informasi pelamar berbasis dokumen CV digital. Metode yang akan digunakan untuk melakukan deteksi pola dalam CV adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas kandidat melalui basis data sehingga harapannya terbentuk sebuah sistem yang dapat mengenali profil pelamar secara lengkap hanya dengan menggunakan CV digital.

## Penjelasan Implementasi

Dalam tugas ini, Anda akan mengembangkan sebuah sistem ATS (Applicant Tracking System) berbasis CV Digital dengan memanfaatkan teknik Pattern Matching. Implementasi sistem ini akan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt (*Aho-Corasick* apabila mengerjakan bonus) untuk menganalisis dan mencocokkan pola dalam dokumen CV digital, sesuai dengan konsep yang telah dipelajari dalam materi dan slide perkuliahan.




**Gambar 2.** Skema Implementasi *Applicant Tracking System*

Sistem ini bertujuan untuk mencocokkan kata kunci dari user terhadap isi CV pelamar kerja dengan pendekatan pattern matching menggunakan algoritma KMP (Knuth-Morris-Pratt) atau BM (Boyer-Moore). Semua proses dilakukan secara in-memory, tanpa menyimpan hasil pencarian—hanya data mentah (raw) CV yang disimpan. Pengguna (HR atau rekruter) akan memberikan input berupa daftar kata kunci yang ingin dicari (misalnya: "python", "react", dan "sql") serta jumlah CV yang ingin ditampilkan (misalnya Top 10 matches). Setiap file CV dalam format PDF akan dikonversi menjadi satu string panjang yang memuat seluruh teks dari dokumen tersebut. Proses konversi ini bertujuan untuk mempermudah pencocokan pola menggunakan algoritma string matching, sehingga setiap keyword dapat dicari secara efisien dalam satu representasi data linear.

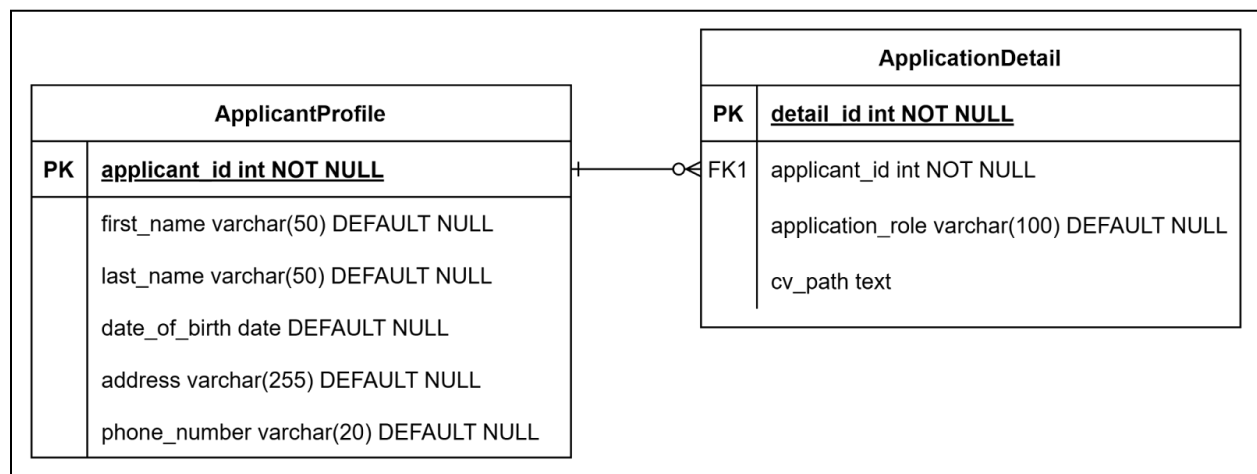
Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

**Tabel 1.** Hasil ekstraksi teks dari CV ATS

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

Pada tahap implementasi ini, setiap CV yang telah dikonversi menjadi string panjang untuk mempermudah proses pencocokan. Representasi ini menjadi dasar dalam mencari CV yang

paling relevan dengan kata kunci yang dimasukkan oleh pengguna. Proses pencarian dilakukan dengan menggunakan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) untuk menemukan CV yang memiliki kemiripan tertinggi dengan kebutuhan yang ditentukan. Apabila tidak ditemukan satupun CV dalam basis data yang memiliki kecocokan kata kunci secara exact match menggunakan algoritma KMP maupun Boyer-Moore, maka sistem akan mencari CV yang paling mirip berdasarkan tingkat kemiripan di atas ambang batas tertentu (threshold). Hal ini mempertimbangkan kemungkinan adanya kesalahan pengetikan (typo) oleh pengguna atau HR saat memasukkan kata kunci. Anda diberikan **keleluasaan untuk menentukan nilai ambang batas persentase** kemiripan tersebut, dengan syarat dilakukan pengujian terlebih dahulu untuk menemukan nilai tuning yang optimal dan **dijelaskan secara rinci dalam laporan**. Metode perhitungan tingkat kemiripan harus diterapkan menggunakan algoritma **Levenshtein Distance**.



**Gambar 3.** Skema basis data CV ATS

Dalam skema basis data ini, tabel **ApplicantProfile** menyimpan informasi pribadi pelamar, sedangkan tabel **ApplicationDetail** menyimpan detail aplikasi yang diajukan oleh pelamar tersebut. Relasi antara tabel **ApplicantProfile** dan **ApplicationDetail** adalah one-to-many, karena seorang pelamar dapat mengajukan lamaran untuk beberapa posisi dalam perusahaan yang sama, atau bahkan perusahaan yang berbeda. Setiap lamaran mungkin memerlukan dokumen yang berbeda, seperti CV yang telah disesuaikan untuk peran tertentu.

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

Atribut **cv\_path** pada tabel **ApplicationDetail** digunakan untuk menyimpan lokasi berkas CV digital pelamar di dalam repositori sistem. Lokasi penyimpanan mengikuti struktur folder di direktori **data/**, sebagaimana dijelaskan dalam struktur *repository* pada bagian [pengumpulan](#)

tugas. Berkas CV yang tersimpan akan dianalisis oleh sistem ATS (Applicant Tracking System) yang dikembangkan dalam Tugas Besar ini.

## Penggunaan Program

**CV Analyzer App**

**Keywords :**  
React, Express, HTML

**Search Algorithm:**  
KMP ☐ BM ☒

**Top Matches:**  
3

**Search**

**Results**  
100 CVs scanned in 100ms

**Farhan** 4 matches  
Matched keywords:  
1. React: 1 occurrence  
2. Express: 2 occurrences  
3. HTML: 1 occurrence  
[Summary](#) [View CV](#)

**Aland** 1 match  
Matched keywords:  
1. React: 1 occurrence  
[Summary](#) [View CV](#)

**Ariel** 1 match  
Matched keywords:  
1. Express: 1 occurrence  
[Summary](#) [View CV](#)

**Gambar 4.** Contoh Antarmuka Program (Halaman *Home*)

**CV Summary**

**Farhan**  
Birthdate: 05-19-2025  
Address: Masjid Salman ITB  
Phone: 0812 3456 7890

**Skills:**  
React Express HTML

**Job History:**  
**CTO**  
2003-2004  
Leading the organization's technology strategies

**Education:**  
**Informatics Engineering** (Institut Teknologi Bandung)  
2022-2026

**Gambar 5.** Contoh Antarmuka Program (Halaman *Summary*)

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

- Judul Aplikasi

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

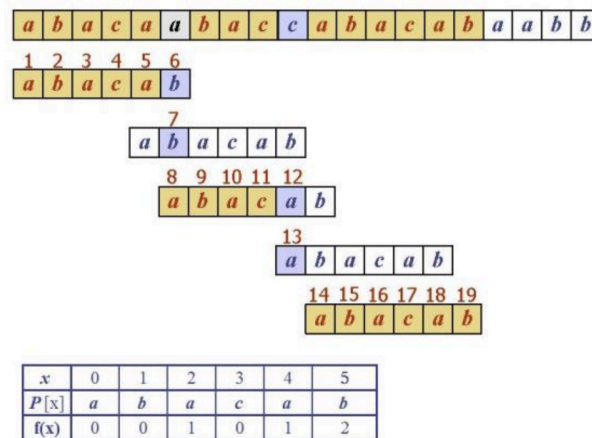
Secara umum, berikut adalah cara umum penggunaan program:

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

## Bab 2 : Landasan Teori

### 2.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt atau KMP dikembangkan oleh Donald Knuth, James H. Morris, dan Vaughan Pratt untuk melakukan pencocokan string secara linier tanpa memeriksa kembali karakter teks yang sudah pernah dibandingkan. KMP mencapai hal itu dengan tahap pra kalkulasi yang membentuk tabel lps (longest proper prefix that is also suffix). Untuk setiap posisi  $j$  dalam pola,  $lps[j]$  memuat panjang awalan pola terpanjang yang juga merupakan akhiran hingga posisi tersebut. Nilai inilah yang menentukan sejauh apa pola dapat digeser pada saat terjadi ketidakcocokan. Proses membangun tabel lps hanya membutuhkan satu kali pemindaian pola sehingga memakan waktu  $O(m)$ ,  $m$  adalah panjang pola.



Sesudah tabel lps tersedia, algoritma memulai pencarian dengan dua indeks:  $i$  menunjuk karakter teks dan  $j$  menunjuk karakter pola. Selama  $text[i]$  sama dengan  $pattern[j]$ , kedua indeks bergerak maju. Jika  $j$  mencapai  $m$ , berarti pola ditemukan dan KMP mencatat posisi kemunculan lalu mengatur  $j$  menjadi  $lps[m-1]$  untuk mencari kemunculan selanjutnya tanpa memundurkan  $i$ . Ketika terdapat ketidakcocokan, ada dua kemungkinan: jika  $j$  bernilai 0 maka  $i$  dinaikkan satu karena pola belum cocok sama sekali, tetapi jika  $j$  lebih besar dari 0 maka  $j$  diatur menjadi  $lps[j-1]$ . Dengan cara itu, indeks  $i$  tidak pernah mundur sehingga setiap karakter teks diproses paling banyak satu kali. Kompleksitas pencarian adalah  $O(n)$ ,  $n$  adalah panjang teks, sehingga total waktu KMP menjadi  $O(n + m)$  dan bersifat linier pada semua kasus.

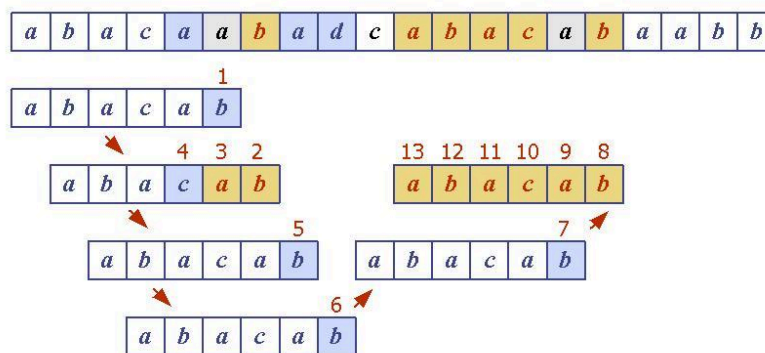
Keunggulan utama KMP terletak pada kestabilan performa yang tidak dipengaruhi distribusi karakter dan pada implementasi yang relatif sederhana. Algoritma ini cocok untuk pola pendek hingga menengah, aliran data waktu nyata, atau situasi yang membutuhkan jaminan waktu proses terburuk tetap linier. Kekurangannya adalah pola sering hanya bergeser



sedikit karena KMP semata-mata mengandalkan hubungan prefiks-sufiks, akibatnya pada pola panjang dengan alfabet besar, KMP dapat kalah cepat dari algoritma heuristik seperti Boyer-Moore. Namun, apabila dibutuhkan metode pencarian yang konsisten, mudah dipahami, dan bebas dari kompleksitas terburuk kuadratik, KMP tetap menjadi pilihan yang kuat dalam pemrosesan teks.

## 2.2 Algoritma Boyer-Moore

Algoritma Knuth-Morris-Pratt (KMP) dirancang untuk mempercepat pencocokan string dengan mengeliminasi pemeriksaan ulang karakter yang telah dibandingkan sebelumnya. Caranya ialah membangun tabel faktor prefiks-sufiks terpanjang (longest proper prefix which is also suffix, lps) pada tahap praproses. Nilai-nilai dalam tabel ini menunjukkan seberapa jauh pola dapat digeser ketika terjadi ketidakcocokan tanpa perlu memundurkan penunjuk pada teks, sehingga setiap karakter teks dipindai paling banyak sekali. Tahap praproses memerlukan waktu  $O(m)$  dengan  $m$  panjang pola, sedangkan tahap pencarian berjalan dalam  $O(n)$  dengan  $n$  panjang teks, totalnya linier  $O(n+m)$ . KMP stabil pada semua input, relatif mudah diimplementasikan, dan cocok untuk pola berukuran pendek hingga sedang maupun alfabet kecil, meskipun kadang menghasilkan lompatan pendek karena tidak memanfaatkan informasi posisi karakter terakhir.



Boyer-Moore (BM) memakai pendekatan berbeda dengan memulai perbandingan dari sisi kanan pola dan menerapkan dua heuristik utama, yaitu bad-character dan good-suffix. Heuristik bad-character memanfaatkan posisi kemunculan terakhir karakter yang menyebabkan ketidakcocokan untuk menentukan seberapa jauh pola dapat digeser, sedangkan heuristik good-suffix memanfaatkan kesesuaian sufiks paling kanan pola untuk melakukan lompatan lebih jauh lagi. Tabel bad-character dibangun dalam  $O(m + |\Sigma|)$  dan tabel good-suffix dalam  $O(m)$ , di mana  $|\Sigma|$  adalah ukuran alfabet. Pada praktiknya BM sering lebih cepat karena lompatan gesernya dapat melampaui panjang pola, menghasilkan kompleksitas rata-rata sub-linier, meski

pada kasus terburuknya dapat mencapai  $O(nm)$ . Implementasinya lebih kompleks, dan keuntungannya paling terasa pada pola panjang serta alfabet besar, untuk pola sangat pendek atau alfabet terbatas, performanya dapat mendekati atau kalah dari KMP.

Secara ringkas, KMP menawarkan kepastian kinerja linier dengan implementasi lebih sederhana, sedangkan Boyer-Moore memberikan kecepatan tinggi di kebanyakan keadaan nyata berkat lompatan besar, namun membutuhkan praproses lebih matang serta struktur kode lebih rumit. Pemilihan algoritma harus mempertimbangkan karakteristik data: sistem real-time atau pola pendek lebih aman memakai KMP, sementara pemrosesan batch berskala besar dan pola panjang biasanya memperoleh keuntungan signifikan dari Boyer-Moore atau varian sederhananya seperti Boyer-Moore-Horspool.

## **Alur Kerja GUI**

### **1. Startup dan Inisialisasi**

Program dimulai dengan koneksi database MySQL dan inisialisasi komponen utama (PatternMatcher, PDFExtractor, cache system). Window utama berukuran 1000x700 pixel ditampilkan dengan styling retro Windows 98 menggunakan color scheme abu-abu classic dan title bar biru gelap.

### **2. Dialog Konfigurasi CV Loading**

Setelah window utama terbuka, muncul dialog konfigurasi (650x600 pixel) yang scrollable dan resizable. Dialog menampilkan total CV dalam database, input spinbox untuk jumlah CV yang akan di-load, quick select buttons (25, 50, 100, 200, 500, All), dan performance guide dengan estimasi waktu loading berdasarkan color coding.

### **3. Progress Loading**

User memilih jumlah CV dan menekan "Load CVs & Start". Sistem menampilkan progress dialog dengan animated dots dan real-time counter "Loaded X/Y CVs...". Loading menggunakan multi-threading untuk optimasi performa dengan ThreadPoolExecutor.

### **4. Search Parameters Section**

Setelah loading selesai, tampil section pencarian dengan: text input untuk keywords (comma-separated), dropdown algoritma (KMP/Boyer-Moore), checkbox case sensitive, spinbox maximum results (1-100), dan horizontal slider similarity threshold (0-100%).

## 5. Action Buttons

Tiga tombol utama dengan styling retro 3D: "Search Database", "Clear Results", dan "Reload CVs". Semua tombol memiliki hover effect dan cursor hand pointer untuk feedback visual.

## 6. Results Display Area

Area hasil menggunakan Canvas dengan scrollbar vertikal untuk menampilkan multiple result cards. Layout menggunakan grid 2 kolom yang responsive dengan hasil pencarian ditampilkan dalam bentuk cards.

## 7. Result Cards

Setiap card memiliki header biru dengan ranking dan score, content area putih dengan matched keywords dalam label buttons, dan dua action buttons: "Summary" dan "Open CV" untuk akses detail dan file PDF.

## 8. Summary Window

Tombol Summary membuka modal window (800x600 pixel) dengan scrollable content yang menampilkan data terstruktur: Personal Information, Technical Skills, Work Experience, dan Education dalam sections terpisah dengan styling konsisten.

## 9. Search Process dan Feedback

Pencarian memberikan real-time feedback dengan loading message, kemudian menjalankan algoritma pattern matching yang dipilih. Performance metrics ditampilkan di status bar dengan format "Found X matches. Exact: Y ms | Fuzzy: Z ms".

## 10. Error Handling

Sistem menangani berbagai error dengan messagebox informatif: warning untuk keywords kosong, "No Results Found" untuk pencarian tanpa hasil, dan error messages untuk masalah database atau file corruption dengan suggestion yang actionable.

## Teknologi dan Library yang Digunakan

- **Tkinter**: Digunakan sebagai alternatif untuk prototipe awal (lihat ATSApp di kode).
- **Pathlib**: Untuk menangani path file secara portabel di berbagai sistem operasi.
- **Webbrowser**: Untuk membuka file PDF dengan aplikasi default.
- **ThreadPoolExecutor**: Untuk memuat teks CV secara paralel saat inisialisasi aplikasi, meningkatkan performa.

## Optimasi dan Fitur Tambahan

- **Preloading CV:** Teks CV diekstrak dan disimpan di cache saat aplikasi dimulai untuk mempercepat pencarian (`_preload_all_cvs` di `ATSTApp`).
- **Pencarian Hibrida:** Menggabungkan pencocokan eksak dan fuzzy untuk hasil yang lebih fleksibel.
- **Scroll Area:** Digunakan di `ResultWidget` dan `SummaryWidget` untuk menangani konten yang panjang.
- **Penanganan Error:** Menampilkan pesan error untuk kasus seperti file PDF tidak ditemukan atau koneksi database gagal.
- **Konfigurasi:** Parameter seperti ukuran jendela, threshold fuzzy, dan direktori data didefinisikan di `config.py` untuk fleksibilitas.

## Bab 3 : Analisis Pemecahan Masalah

### 3.1 Langkah-langkah pemecahan masalah

Langkah dalam memecahkan masalahnya adalah berikut, sesuai dengan jalur program yang kami buat, program lebih dulu membuat objek `DatabaseConnection` dan memanggil `connect`. Pemanggilan ini memakai parameter `host`, `user`, dan `password` yang disimpan pada `config.py`. Jika koneksi ke mesin `MySQL` atau `MariaDB` gagal karena kredensial salah, port tertutup, atau layanan basis data belum berjalan, aplikasi langsung memanggil `sys.exit(1)`. Pola keluar dini ini penting karena mencegah Tkinter memulai event loop dalam keadaan basis data tidak tersedia, kegagalan dipindahkan ke tahap awal sehingga mudah dilacak serta tidak menimbulkan pesan kesalahan berantai di lapisan antarmuka. Begitu koneksi berhasil, fungsi `useDatabase` memilih skema yang telah didefinisikan pada variabel `DATABASE_CONFIG["database"]`. Langkah ini memastikan semua query selanjutnya terarah ke tabel yang benar tanpa perlu menyisipkan perintah `USE` di setiap statement.

Sesudah fondasi basis data mantap, kode membuat window Tkinter lewat `Tk()` dan menyerahkannya ke konstruktor `ATSApp`. Di dalam konstruktor, properti penting diinisialisasi: `self.matcher` menyimpan instance `PatternMatcher` yang berisi implementasi KMP, Boyer-Moore, serta Levenshtein, `self.extractor` menyimpan `PDFExtractor` yang mendekode halaman PDF menjadi string, `self.cache` berupa kamus kosong yang nantinya berisi pasangan path dengan teks CV. Fungsi `_build_ui` kemudian dijalankan. Pada tahap ini semua komponen antarmuka dideklarasikan secara imperatif menggunakan widget bawaan Tkinter. Header memakai font tebal 18 poin agar judul aplikasi terlihat jelas. Bagian kontrol pencarian disusun dalam Frame bernama `params` dengan grid layout, sehingga label, entry, combobox, spinbox, dan slider tertata rapi tanpa harus menggunakan absolute positioning. Pembuatan kontrol juga menetapkan variabel kelas `StringVar` dan `IntVar`, yang memudahkan pembacaan nilai input tanpa harus mengambil teks dari widget secara manual setiap kali pencarian dilakukan.

Sebelum pengguna melakukan apa-apa, metode `_preload_all_cvs` diaktifkan secara otomatis. Fungsi ini memanggil `get_all_cv_paths` yang mengeksekusi sebuah query `SELECT` ke tabel `ApplicationDetail` untuk mengambil seluruh kolom `cv_path`. Agar pemuatan dokumen tidak menghambat tampilan, pustaka `concurrent.futures` dipakai dengan empat thread. Setiap thread memanggil `PDFExtractor.PDFExtractForMatch`, yaitu rutinitas yang membuka file PDF, menggabungkan seluruh halaman, menormalisasi baris, serta membuang karakter kosong. Nama file pada kolom basis data dicantumkan relatif terhadap folder data, sedangkan variabel global `DATA_DIR` menunjuk root penyimpanan di luar proyek, sehingga path absolut dibangun dengan `Path(DATA_DIR) / Path(p).relative_to("data")`. Apabila `PDFExtractor` gagal memproses file karena PDF terenkripsi, rusak, atau tidak terbaca, blok `try/except` mendaftarkan value string kosong pada `self.cache`. Mekanisme ini membuat satu file bermasalah tidak menjatuhkan loop pramuat, dan pengguna tetap bisa mencari di korpus lainnya. Waktu total pramuat dihitung

menggunakan `time.time()` sebelum dan sesudah proses sehingga pengembang dapat memantau kinerja I/O.

Pada saat pengguna menekan tombol Search CV, method `on_search` membersihkan daftar hasil lama dengan memanggil `destroy` pada setiap widget anak di `self.results_frame`. Kata kunci diambil dari entry lalu dipecah dengan koma, dipangkas spasi, dan dikonversi menjadi daftar string. Combobox algoritma dikonversi ke nilai internal “KMP” atau “BM”. Slider fuzzy diubah menjadi bilangan desimal antara nol dan satu. Program kemudian melakukan iterasi atas kamus `self.cache`. Untuk setiap dokumen, `exactMatch` dipanggil lebih dahulu karena pencocokan presisi dinilai lebih tinggi di skor. `exactMatch` mengembalikan struktur yang berisi dictionary kata kunci ke jumlah kemunculan, serta waktu eksekusi dalam satuan milidetik yang sudah diubah menjadi float guna dijumlahkan. Jika jumlah kemunculan presisi lebih besar dari nol, skor dihitung sebagai total kemunculan dikalikan seribu. Perkalian besar ini sengaja diberlakukan agar CV yang cocok presisi otomatis berada di urutan lebih tinggi dibanding CV yang hanya lolos fuzzy.

Apabila pencarian presisi menemukan nol kecocokan, fungsi `fuzzyMatch` dipanggil. Di sini string dibandingkan memakai jarak Levenshtein yang diimplementasi `PatternMatcher`. Algoritma ini menghitung edit distance antara kata kunci dan token dalam teks CV lalu membaginya dengan panjang kata terpanjang untuk mendapatkan rasio kemiripan. Bila rasio di atas ambang slider, kata dianggap cocok. Fuzzy cocok memberikan skor yang setara dengan total kemunculan, tanpa faktor pengali seribu, sehingga tetap bisa muncul di daftar tetapi kalah prioritas dari match presisi. Waktu eksekusi exakt dan fuzzy untuk setiap dokumen disimpan di variabel `total_ex` dan `total_fu`, kemudian ditampilkan di label di bawah kontrol pencarian agar pengguna paham beban komputasi yang terjadi.

Setelah seluruh dokumen diproses, daftar hits disortir berdasarkan skor secara menurun dan hanya `top_n` pertama yang dipertahankan. Untuk setiap path di hits, aplikasi memanggil `get_summary_data_by_cv_path`. Fungsi ini menjalankan query join antara tabel `ApplicantProfile` dan `ApplicationDetail` sehingga hanya satu perjalanan ke server diperlukan untuk memuat nama, tanggal lahir, nomor telepon, dan lokasi PDF. Fungsi internal extractor regex kemudian dijalankan untuk mengambil blok teks skill, pengalaman kerja, dan pendidikan langsung dari string CV yang sudah ada di cache, bukan membuka PDF lagi, sehingga penggunaan memori dan waktu lebih efisien. Data kembalian kemudian ditempatkan ke objek `ResultCard`. Objek ini hanyalah struktur data sederhana yang menyimpan nama lengkap, path, map keyword, dan total kecocokan, membuat logika tampilan terpisah dari detail data.

Frame hasil diberikan border raised dan padding agar terlihat seperti kartu. Dalam satu kartu, nama kandidat dipasang dengan font 14 tebal. Jika ada keyword yang cocok, label menampilkan daftar format kata(frekuensi) dipisahkan koma. Total kecocokan juga ditampilkan. Dua tombol diletakkan di sisi kanan bawah kartu: Summary dan View CV. Tombol Summary memanggil `show_summary` yang membangun jendela baru `Toplevel`. Jendela ini menampilkan

header gelap dengan teks putih, kemudian bagian personal info, daftar chip skill sebagai label dengan border ridge, kotak abu muda untuk setiap pengalaman kerja, dan kotak serupa untuk setiap entri pendidikan. Pemakaian wraplength memastikan teks panjang dibungkus sehingga tidak akan melebar melebihi 560 piksel. Jendela diatur modal dengan win.transient dan win.grab\_set supaya pengguna tidak bisa berinteraksi dengan jendela utama sebelum menutup ringkasan.

Tombol View CV menjalankan open\_cv. Fungsi ini mengubah path relatif ke path absolut memakai rumus sama dengan pramuat. Jika file ada, browser bawaan sistem dipanggil untuk membuka PDF sehingga pengguna dapat melihat format asli dokumen. Bila file hilang atau dipindahkan, dialog kesalahan muncul lewat messagebox.showerror. Penanganan kesalahan di sini mencegah crash dan memberikan pesan yang bisa dimengerti pengguna akhir.

Setelah pengguna selesai meninjau hasil dan menutup jendela utama, root.mainloop() selesai, lalu db.disconnect() dipanggil untuk memutus koneksi basis data. Dengan demikian semua sumber daya koneksi socket MySQL, thread executor, handle file dibebaskan secara teratur. Rangkaian langkah tersebut memastikan aplikasi bekerja dari awal hingga akhir: koneksi validasi, pemuatan PDF multithread, pencarian presisi dan fuzzy, pengambilan metadata SQL, ekstraksi ringkas dengan regex, penyajian hasil di GUI, dan penanganan file PDF, semuanya dilaksanakan dalam urutan jelas yang memungkinkan pengembang melakukan debug di setiap tahap apabila terjadi kesalahan.

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

Proses yang diselesaikan adalah mencari kemunculan pola (pattern) tertentu dalam teks yang panjang, misalnya mencari kata kunci dalam dokumen CV untuk keperluan Applicant Tracking System (ATS). Tujuannya adalah menemukan semua posisi di mana pola muncul dalam teks secara efisien, baik secara eksak (tepat sama) maupun fuzzy (mirip dengan toleransi kesalahan). Untuk itu, kami memilih dua algoritma: KMP dan BM, karena keduanya dikenal cepat untuk pencocokan eksak, dan kami juga mengintegrasikan Levenshtein Distance untuk pencocokan fuzzy.

Untuk masalah pencocokan pola string membutuhkan:

- **Teks:** Dokumen CV yang sudah diekstrak menjadi string.
- **Pola:** Kata kunci (misalnya "Python", "Java") yang dicari.
- **Output:** Daftar posisi di mana pola ditemukan dalam teks.

Untuk KMP menggunakan *failure function* (atau disebut LPS - Longest Proper Prefix which is also Suffix) untuk menghindari perbandingan ulang karakter yang sudah diketahui tidak cocok. Kami memetakan elemen-elemen masalah ke dalam langkah-langkah KMP sebagai berikut:

### 1. Pra-pemrosesan Pola (`computeFailureFunction`):

- Tujuan: Membuat array LPS untuk pola, yang menyimpan panjang prefix terpanjang yang juga merupakan suffix untuk setiap posisi dalam pola.
- Pemetaan: Misalnya, untuk pola "ABACABC", Kami menghitung array LPS seperti [0, 0, 1, 0, 1, 2, 0]. Ini membantu KMP "melompat" ke posisi yang tepat saat terjadi kegagalan pencocokan, tanpa harus memeriksa ulang karakter yang sudah cocok.
- Implementasi: Dalam metode `computeFailureFunction`, Kami membandingkan karakter pola secara berurutan dan memperbarui array LPS berdasarkan kecocokan prefix dan suffix. Jika karakter tidak cocok, Kami kembali ke LPS sebelumnya untuk mencari prefix yang lebih pendek.

### 2. Pencarian Utama (`search`):

- Tujuan: Mencari semua kemunculan pola dalam teks.
- Pemetaan: Kami menggunakan dua iterator: *i* untuk teks dan *j* untuk pola. Jika karakter cocok, keduanya maju. Jika tidak cocok, Kami gunakan LPS untuk mengatur ulang *j* tanpa memundurkan *i*, sehingga menghemat waktu.
- Implementasi: Dalam metode `search`, Kami menyimpan posisi awal pola yang cocok dalam daftar `result`. Kami juga menangani opsi `caseSensitive` dengan mengonversi teks dan pola ke huruf kecil jika diperlukan.

### 3. Pencarian Banyak Pola (`searchMultiple`):

- **Tujuan:** Mencari beberapa kata kunci sekaligus.
- **Pemetaan:** Kami iterasi setiap pola dalam daftar kata kunci dan panggil `search` untuk masing-masing, menyimpan hasil dalam dictionary.
- **Implementasi:** Metode `searchMultiple` mengembalikan dictionary dengan pola sebagai kunci dan daftar posisi sebagai nilai.

## Tantangan dan Solusi

- **Tantangan:** Memahami bagaimana LPS bekerja untuk pola dengan banyak pengulangan (misalnya "AAA"). Kami harus memastikan bahwa LPS dihitung dengan benar untuk kasus seperti ini.
- **Solusi:** Kami membuat kode `computeFailureFunction` dengan logika iteratif yang jelas, memastikan bahwa setiap karakter dibandingkan dengan benar dan memperbarui `length` (panjang prefix yang cocok) dengan tepat.

Untuk Boyer Moore, sama seperti KMP, tapi punya pendekatan berbeda dengan memanfaatkan dua heuristik: *bad character* dan *good suffix*. Ini cocok untuk teks panjang seperti CV karena bisa melompati banyak karakter. BM bekerja dari kanan ke kiri dalam pola dan menggunakan dua tabel untuk lompatan:



- **Bad Character Table:** Menyimpan posisi terakhir kemunculan setiap karakter dalam pola.
- **Good Suffix Table:** Menentukan seberapa jauh lompatan jika sebagian pola sudah cocok.

Kami memetakan elemen-elemen masalah ke dalam langkah-langkah BM sebagai berikut:

1. **Pra-pemrosesan Bad Character (`preprocessBadCharacter`):**

- **Tujuan:** Membuat tabel yang menyimpan posisi terakhir setiap karakter dalam pola.
- **Pemetaan:** Misalnya, untuk pola "ABC", tabelnya adalah {A: 0, B: 1, C: 2}. Jika karakter di teks tidak cocok, Kami gunakan tabel ini untuk melompat ke posisi yang mungkin cocok.
- **Implementasi:** Dalam metode `preprocessBadCharacter`, Kami iterasi pola dan menyimpan indeks terakhir setiap karakter dalam dictionary bad.

2. **Pra-pemrosesan Good Suffix (`preprocessGoodSuffix`):**

- **Tujuan:** Menghitung seberapa jauh lompatan berdasarkan suffix yang cocok.
- **Pemetaan:** Ini lebih kompleks karena melibatkan dua langkah:
  - Menghitung *border array* untuk menemukan suffix yang cocok.
  - Mengisi tabel good untuk menentukan lompatan berdasarkan suffix atau prefix.
- **Implementasi:** Dalam metode `preprocessGoodSuffix`, Kami hitung border array secara mundur dan isi tabel good untuk setiap posisi dalam pola.

3. **Pencarian Utama (`search`):**

- **Tujuan:** Mencari pola dalam teks dengan melompati karakter sebanyak mungkin.
- **Pemetaan:** Kami mulai dari kanan pola, bandingkan dengan teks, dan gunakan tabel bad dan good untuk melompat jika ada ketidakcocokan. Jika pola cocok, Kami catat posisinya.
- **Implementasi:** Dalam metode `search`, Kami gunakan `s` sebagai posisi awal pencocokan dan lompat berdasarkan  $\max(\text{char\_shift}, \text{suffix\_shift})$ . Kami juga menangani caseSensitive seperti pada KMP.

4. **Pencarian Banyak Pola (`searchMultiple`):**

- **Tujuan:** Sama seperti KMP, mencari beberapa kata kunci.
- **Pemetaan:** Kami iterasi setiap pola dan panggil `search`, menyimpan hasil dalam dictionary.
- **Implementasi:** Metode `searchMultiple` mirip dengan KMP, tapi menggunakan logika BM untuk pencarian.

## Tantangan dan Solusi

- **Tantangan:** Memahami dan mengimplementasikan *good suffix* karena logikanya lebih rumit dibandingkan *bad character*.
- **Solusi:** Kami pecah prosesnya menjadi dua langkah (border array dan pengisian tabel *good*) dan uji coba dengan pola sederhana untuk memastikan lompatan benar. Kami juga gunakan logika max untuk memilih lompatan terbesar antara *bad character* dan *good suffix*.

## Integrasi dalam Aplikasi ATS

Dalam konteks ATS, kedua algoritma diintegrasikan melalui kelas `PatternMatcher`:

- **exactMatch:** Memilih antara KMP atau BM untuk pencocokan eksak berdasarkan input pengguna.
- **fuzzyMatch:** Menggunakan Levenshtein Distance untuk mencari kata kunci yang mirip (misalnya, "pythun" cocok dengan "Python").
- **hybridMatch:** Menggabungkan pencocokan eksak (KMP/BM) dan fuzzy untuk hasil yang lebih lengkap.

## Pemetaan ke Aplikasi:

- Saat program dijalankan, seluruh teks CV akan di-preload. Teks CV diekstrak menggunakan `PDFExtractor` dan dibersihkan dengan `RegexExtractor`.
- Kata kunci dari pengguna diproses oleh `PatternMatcher` untuk mencari kecocokan.
- Hasil disimpan dalam memori dan ditampilkan melalui GUI.

## 3.3 Fitur fungsional dan arsitektur aplikasi web yang dibangun

Berikut adalah fitur-fitur utama yang ada di aplikasi ATS, berdasarkan kode yang diberikan:

### 1. Pencarian Kata Kunci pada CV

Pengguna dapat memasukkan kata kunci (misalnya "Python", "Java") untuk mencari CV yang relevan. Kata kunci dipisahkan dengan koma.

#### Implementasi:

- Pengguna memasukkan kata kunci melalui widget pencarian (`SearchWidget`) di GUI.
- Aplikasi mendukung dua algoritma pencocokan eksak: KMP dan BM, yang dipilih melalui dropdown di `SearchWidget`.
- Untuk kata kunci yang tidak cocok secara eksak, aplikasi menggunakan Levenshtein Distance untuk pencocokan fuzzy dengan ambang batas (*threshold*) yang dapat diatur melalui slider.

- Hasil pencarian ditampilkan dalam bentuk kartu hasil (ResultWidget), menunjukkan nama pelamar, jumlah kecocokan, dan kata kunci yang ditemukan.

**Contoh Penggunaan:** Pengguna memasukkan "Python, SQL", memilih algoritma KMP, dan mengatur *threshold* 70%. Aplikasi akan menampilkan CV yang mengandung "Python" atau "SQL" (atau kata mirip seperti "pythun") dengan skor relevansi.

## 2. Tampilan Hasil Pencarian

Hasil pencarian ditampilkan dalam daftar yang dapat di-scroll, dengan informasi ringkas tentang setiap CV.

### Implementasi:

- Setiap hasil ditampilkan sebagai kartu (ResultCard) yang berisi nama pelamar, jumlah total kecocokan, dan kata kunci yang ditemukan.
- Pengguna dapat mengurutkan hasil berdasarkan skor relevansi (kecocokan eksak memiliki prioritas lebih tinggi).
- Kartu hasil memiliki tombol untuk melihat ringkasan CV atau membuka file CV asli.

**Contoh Penggunaan:** Setelah pencarian, pengguna melihat daftar 10 CV teratas dengan nama pelamar dan jumlah kecocokan (misalnya, "John Doe: 5 kecocokan").

## 3. Ringkasan CV

Pengguna dapat melihat ringkasan informasi dari CV yang dipilih, seperti nama, tanggal lahir, nomor telepon, keterampilan, pengalaman kerja, dan pendidikan.

### Implementasi:

- Ringkasan ditampilkan melalui window modal Toplevel dengan scrollable content, yang diisi dengan data dari `get_summary_data_by_cv_path()`. Informasi diambil langsung dari database dan ditampilkan dalam sections terstruktur (Personal Information, Technical Skills, Work Experience, Education) menggunakan `method_build_retro_summary()`.
- Pengguna dapat mengklik tombol "View CV" untuk membuka file PDF asli.

**Contoh Penggunaan:** Pengguna memilih CV John Doe, lalu melihat ringkasan berisi "Nama: John Doe, Keterampilan: Python, SQL, Pengalaman: Software Engineer di ABC Corp".

## 4. Penampilan CV Asli

Pengguna dapat membuka file PDF CV asli untuk melihat dokumen lengkap.

### Implementasi:

- Fitur ini diimplementasikan melalui method `open_cv()` di class `ATSApp`, yang menggunakan `webbrowser.open_new()` untuk membuka file PDF menggunakan aplikasi default sistem. Path file direkonstruksi dari `DATA_DIR` dan `cv_path` relatif.
- Path ke file PDF diambil dari database (`ApplicationDetail.cv_path`) dan diakses melalui fungsi `open_cv` di `ATSApp`.

**Contoh Penggunaan:** Pengguna mengklik tombol "View CV" pada kartu hasil, dan file PDF CV terbuka di browser atau aplikasi PDF.

## 5. Manajemen Data Pelamar

Aplikasi menggunakan `DatabaseConnection` untuk koneksi database dan `src.database.queries` untuk operasi data (`get_all_cv_paths`, `get_summary_data_by_cv_path`). Struktur tabel dan operasi CRUD tidak terlihat dalam `main.py` - hanya query functions yang digunakan untuk mengambil data CV dan summary.

### Implementasi:

- Database → `get_all_cv_paths()` → CV loading dialog
- User selection → Multi-threaded PDF extraction → Cache
- Search → Pattern matching pada cached text → Result display
- Result interaction → Summary view atau PDF opening

### Contoh Penggunaan:

Admin dapat menjalankan `DataSeeder.generateSampleApplicants(30)` untuk membuat 30 data pelamar dummy dengan CV acak.

## 6. Preloading CV

Aplikasi memuat teks dari semua CV di awal untuk mempercepat proses pencarian.

### Implementasi:

- Fungsi `_preload_all_cvs` di `ATSApp` menggunakan `ThreadPoolExecutor` untuk mengekstrak teks dari file PDF secara paralel.
- Teks yang diekstrak disimpan dalam cache (`self.cache`) untuk menghindari ekstraksi ulang saat pencarian.

**Contoh Penggunaan:** Saat aplikasi mulai, semua CV di database diproses menjadi teks, sehingga pencarian berjalan lebih cepat.

## Arsitektur Aplikasi

Aplikasi ATS dirancang dengan arsitektur berlapis (*layered architecture*) untuk memisahkan tanggung jawab dan memudahkan pengembangan serta pemeliharaan. Berikut adalah lapisan utama arsitektur berdasarkan kode:

## 1. Lapisan Presentasi (GUI)

- **Komponen:** Antarmuka pengguna berbasis Tkinter, termasuk ATSAApp (main class), CVLoadDialog, RetroStyle theming, dan integrated components dalam single window architecture.
- **Fungsi:**
  - Menyediakan antarmuka untuk input pengguna (kata kunci, algoritma, *threshold*).
  - Menampilkan hasil pencarian dan ringkasan CV.
  - Membuka file CV asli.
- **Implementasi:**
  - ATSAApp sebagai main application controller yang mengintegrasikan semua komponen GUI dalam single window architecture.
  - Search parameters terintegrasi dalam main window dengan Entry widget untuk keywords, Combobox untuk algorithm selection, dan Scale untuk similarity threshold.
  - Results display menggunakan scrollable Canvas dengan dynamic result cards yang memiliki embedded action buttons untuk Summary dan Open CV.
  - CV Summary ditampilkan melalui modal Toplevel window dengan scrollable content area menggunakan method `_build_retro_summary()`.
- **Alur Kerja:**
  - Pengguna memasukkan kata kunci di search parameters section.
  - Button callback `on_search()` dipicu, yang memulai pencarian melalui PatternMatcher.
  - Hasil ditampilkan di scrollable results area dengan result cards, dan summary dalam modal window jika tombol Summary diklik.

## 2. Lapisan Bisnis (Logika Aplikasi)

- **Komponen:** Kelas PatternMatcher, PDFExtractor, RegexExtractor, ATSAApp.
- **Fungsi:**
  - Mengelola logika pencocokan pola (KMP, BM, Levenshtein).
  - Mengekstrak dan membersihkan teks dari file PDF.
  - Mengkoordinasikan pencarian dan pengolahan hasil.
- **Implementasi:**

- PatternMatcher mengintegrasikan algoritma KMP, BM, dan Levenshtein untuk pencocokan eksak dan fuzzy.
- PDFExtractor mengekstrak teks dari PDF menggunakan pypdf.
- RegexExtractor membersihkan teks dan mengekstrak bagian CV seperti keterampilan, pengalaman, dan pendidikan.
- ATSApp mengatur alur kerja utama, termasuk *preloading* CV dan pencarian.
- **Alur Kerja:**
  - Teks CV diekstrak oleh PDFExtractor dan dibersihkan oleh RegexExtractor.
  - PatternMatcher memproses kata kunci untuk mencari kecocokan.
  - Hasil dikembalikan ke ATSApp untuk ditampilkan di GUI.

### 3. Lapisan Data (Database)

- **Komponen:** Kelas DatabaseConnection, DatabaseSchema, DataSeeder, model data (ApplicantProfile, ApplicationDetail).
- **Fungsi:**
  - Menyimpan dan mengelola data pelamar dan aplikasi.
  - Menyediakan akses ke file CV melalui path yang tersimpan.
  - Mendukung *seeding* data untuk pengujian.
- **Implementasi:**
  - DatabaseConnection menangani koneksi ke MySQL dan operasi query.
  - DatabaseSchema mendefinisikan struktur tabel (ApplicantProfile dan ApplicationDetail).
  - DataSeeder menghasilkan data dummy menggunakan Faker dan file PDF acak.
  - Model ApplicantProfile dan ApplicationDetail memetakan data ke struktur database.
- **Alur Kerja:**
  - Data pelamar disimpan saat *seeding* atau input manual.
  - Query seperti `get_summary_data_by_cv_path` mengambil data untuk ringkasan CV.
  - Path CV diakses untuk ekstraksi teks atau penampilan file PDF.

### 4. Lapisan Utilitas

- **Komponen:** Modul config, fungsi database queries (`get_all_cv_paths`, `get_summary_data_by_cv_path`), RetroStyle class untuk theming, dan CVLoadDialog untuk utility functions.
- **Fungsi:**
  - Menyediakan konfigurasi (misalnya, `DATABASE_CONFIG`, `DATA_DIR`).

- Menyediakan fungsi utility untuk GUI (RetroStyle theming, dialog creation).
- Menyediakan query functions untuk akses database yang sudah siap pakai.
- **Implementasi:**
  - Config menyimpan pengaturan seperti koneksi database dan direktori data.
  - Database queries (get\_all\_cv\_paths, get\_summary\_data\_by\_cv\_path) digunakan untuk mengambil data CV dan summary dari database.
  - RetroStyle class menyediakan color constants dan styling guidelines untuk konsistensi UI theme.
  - CVLoadDialog berfungsi sebagai utility component untuk CV loading configuration dengan responsive design.
  - Helper methods seperti \_create\_retro\_button(), \_create\_3d\_frame() untuk UI component creation.

### 3.4 Contoh ilustrasi kasus

Terdapat beberapa kasus yang mungkin dalam aplikasi ini:

#### 1. Pencocokan Utuh Kata Kunci yang Normal

- **Deskripsi:** Kata kunci yang dimasukkan pengguna (misalnya, "Python, SQL") sesuai persis dengan teks dalam CV tanpa modifikasi.
- **Proses:** Algoritma KMP atau Boyer-Moore digunakan untuk pencocokan eksak, menghasilkan kecocokan 100% karena pola kata kunci sama dengan teks CV.
- **Hasil:** Sistem mendeteksi semua kata kunci dengan akurasi penuh, dan CV langsung ditampilkan sebagai hasil relevan tinggi.
- **Contoh:** Pengguna mencari "Python, SQL" pada CV "Rina Sari" yang berisi "Saya mahir Python dan SQL", menghasilkan similarity 100%, dan CV diterima tanpa masalah.

#### 2. Pencocokan Utuh Kata Kunci yang Altered

- **Deskripsi:** Kata kunci yang dimasukkan mengalami perubahan kecil (misalnya, kesalahan ketik seperti "Pythn" atau "SQL"), tetapi teks CV tetap utuh.
- **Proses:** Algoritma Levenshtein Distance digunakan untuk pencocokan fuzzy, dengan similarity sekitar 90-95% jika perubahan kecil (misalnya, 1-2 karakter berbeda). Sistem menerapkan threshold (misalnya, 70%) untuk menentukan kecocokan.
- **Hasil:** Jika similarity di atas ambang batas, CV tetap dianggap relevan, tetapi dengan peringatan untuk verifikasi manual. Jika di bawah, hasil dikecualikan.
- **Contoh:** Pengguna mengetik "Pythn, SQL" pada CV "Budi Santoso" yang berisi "Python dan SQL", menghasilkan similarity 92%, dan CV diterima setelah pengecekan manual.

### 3. Pencocokan Utuh Kata Kunci yang Crop

- **Deskripsi:** Teks CV hanya sebagian tersedia untuk dicocokkan (misalnya, hanya 50% teks diekstrak karena format PDF rusak), sehingga data kata kunci terpotong.
- **Proses:** Algoritma mencoba mencocokkan kata kunci pada teks yang tersedia, tetapi similarity rendah (misalnya, 60-70%) karena data tidak lengkap.
- **Hasil:** Sistem gagal mendeteksi kecocokan penuh, menyarankan pengguna untuk mengunggah ulang CV atau menggunakan dokumen alternatif.
- **Contoh:** CV "Ani Wijaya" hanya mengekstrak "Python" dari "Python, SQL, Excel" karena kerusakan PDF, menghasilkan similarity 70%, dan proses dihentikan untuk pengulangan.

### 4. Pencocokan Utuh Kata Kunci yang Rotate

- **Deskripsi:** Teks CV diekstrak dengan rotasi atau kesalahan orientasi (misalnya, teks miring karena pemindaian buruk), tetapi isi utuh dan dapat diperbaiki.
- **Proses:** Sistem menyesuaikan orientasi teks menggunakan teknik prapemrosesan (jika tersedia) dan mencocokkan kata kunci, menghasilkan similarity mendekati 100% setelah koreksi.
- **Hasil:** CV diterima dengan sukses, menunjukkan ketahanan sistem terhadap variasi pemindaian.
- **Contoh:** CV "Maria Lee" diekstrak dengan teks miring, tetapi setelah koreksi, "Python, SQL" terdeteksi dengan similarity 98%, dan verifikasi berhasil.



## Bab 4 : Implementasi dan Pengujian

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

#### 4.1.1 Struktur Data

##### **Tabel ApplicantProfile**

Menyimpan informasi personal pelamar. applicant\_id (INT, PK, AUTO\_INCREMENT) menjamin baris unik. Kolom lain: first\_name, last\_name, date\_of\_birth, address, phone\_number. Data dipanggil di ringkasan (fungsi show\_summary).

##### **Tabel ApplicationDetail**

Menyimpan riwayat lamaran. application\_id adalah kunci utama, sedangkan applicant\_id menjadi foreign key yang menghubungkan ke ApplicantProfile. Kolom cv\_path memuat lokasi PDF di disk, dan application\_role menyimpan jabatan yang dilamar. Menyimpan riwayat lamaran. application\_id (INT, PK) dan applicant\_id (INT, FK ke ApplicantProfile). cv\_path menyimpan jalur PDF relatif, application\_role posisi yang dilamar.

##### **ResultCard (Python class)**

Objek ringan yang dipakai di lapisan GUI. Ia hanya memuat full\_name, cv\_path, dictionary matched\_keywords, dan total\_matches. Dengan kontainer ini, logika tampilan dapat mengambil semua data yang diperlukan dari satu sumber. Digunakan saat membangun kartu hasil di canvas Tkinter.

##### **SummaryData (Python class)**

Menyimpan data yang lebih lengkap: nama lengkap, tanggal lahir, nomor telepon, list skills, list work\_experience, list education, serta path PDF. Objek ini dihasilkan oleh fungsi get\_summary\_data\_by\_cv\_path dan dipakai di jendela ringkasan.

##### **Cache CV di memori**

Kelas ATSApp memiliki self.cache, sebuah dictionary yang memetakan path PDF ke string teks CV. Pemuatan awal dokumen dimasukkan ke cache agar proses pencarian tidak perlu membaca file berulang-ulang.

##### **Konstanta konfigurasi**

DATABASE\_CONFIG berisi host, port, user, password, dan nama basis data. DATA\_DIR menunjuk

folder penyimpanan dokumen PDF. Variabel ini dipakai di banyak fungsi path builder agar konsisten.

#### **4.1.2 Moduk dan Fungsi**

##### **Koneksi dan Akses Data**

DatabaseConnection.connect() membuka koneksi MySQL dan membuat cursor buffered. execute(sql, params=None, commit=False) satu pintu semua perintah, bila commit True, langsung conn.commit(). queries.get\_all\_cv\_paths(conn) memanggil SELECT cv\_path FROM ApplicationDetail. queries.get\_summary\_data\_by\_cv\_path(conn, path) melakukan JOIN, membentuk SummaryData, dan mengambil teks CV (cache dulu, disk jika perlu).

- DatabaseConnection.connect()  
membuka koneksi MySQL dan membuat cursor buffered.
- execute(sql, params=None, commit=False)  
satu pintu semua perintah, bila commit True, langsung conn.commit().
- queries.get\_all\_cv\_paths(conn)  
memanggil SELECT cv\_path FROM ApplicationDetail.
- queries.get\_summary\_data\_by\_cv\_path(conn, path) melakukan JOIN, membentuk SummaryData, dan mengambil teks CV (cache dulu, disk jika perlu).

##### **Ekstraksi PDF**

PDFExtractor.PDFtoText membaca seluruh halaman PDF memakai pustaka PyPDF dan menggabungkannya menjadi satu string berdelimiter newline. PDFExtractForMatch memanggil fungsi cleansing sehingga string siap dimasukkan ke algoritma pencarian (tanpa karakter aneh, tab, atau newline ganda).

- PDFExtractor.PDFtoText(path)  
Membuka PDF via pypdf.PdfReader. Loop setiap halaman, gabung teks dengan "\n".join(...).
- PDFExtractor.PDFExtractForMatch(path)  
Memanggil PDFtoText. Memanggil RegexExtractor.cleansText untuk meratakan spasi, tab, karakter non-ASCII. Mengembalikan string satu baris siap di-scan algoritma.

##### **Ekstraksi Informasi dengan Regex**

RegexExtractor.cleanseText menormalisasi teks (huruf besar-kecil, spasi ganda, simbol non-ASCII). extract\_cv\_sections mendeteksi header fleksibel seperti Skills, Work History, Education, lalu memecah masing-masing bagian menjadi list bullet atau kalimat. Fungsi ini memastikan hasil ringkasan rapi walau format CV berbeda-beda.

- RegexExtractor.cleanseText(text)  
lower() opsional, hapus \r, tab, dan spasi ganda.
- RegexExtractor.extract\_cv\_sections(text)  
Cari header fleksibel (Skills, Work History, Experience, Education). Tangkap baris di bawah header sampai header berikut atau EOF. Kembalikan dict: skills, work\_experience, education berupa list string.

### Algoritma Pencarian Pola

Tiga file terpisah di folder algorithm memuat implementasi KMP, Boyer-Moore, dan perhitungan Levenshtein. PatternMatcher.exactMatch memilih salah satu algoritma presisi, mengulang masukan keyword, dan mengembalikan hitungan kemunculan. fuzzyMatch menjalankan Levenshtein terhadap kata kunci yang gagal ditemukan, lalu mengecek ratio kesamaan dengan threshold pengguna.

- **KnuthMorrisPratt.py**  
ComputeFailureFunction(p) buat array LPS  $O(m)$ . search(t, p) geser index j ke lps saat mismatch,  $O(n)$ .
- **BoyerMoore.py**  
PreprocessBadCharacter(p) tabel kemunculan terakhir tiap char.  
preprocessGoodSuffix(p) tabel geser suffix. search(t, p) banding dari kanan, geser max(bad, good).
- **LevenshteinDistance.py**  
Calculate(a, b) matriks edit distance  $O(mn)$ . similarity(a, b) hitung rasio kemiripan.
- **PatternMatcher**  
ExactMatch(text, keywords, algorithm) pilih KMP atau BM, hitung posisi, waktu eksekusi.  
FuzzyMatch(text, keywords, threshold) jalankan similarity untuk token teks yang belum cocok.

### Antarmuka Pengguna

ATSApp.\_build\_ui merakit seluruh widget: entry keywords, combobox algoritma, spinbox Top N, slider threshold fuzzy, label waktu, dan canvas hasil dengan scrollbar. \_preload\_all\_cvs

mengambil daftar path dari database, lalu menjalankan ekstraksi PDF secara paralel dengan empat thread. Outputnya dimasukkan ke cache. `on_search` membaca input, memproses pencarian di setiap CV, mencatat waktu, menyortir hasil, lalu membuat kartu tampilan. `show_summary` membangun jendela detail yang menampilkan data dari `SummaryData`. `open_cv` mengonversi path relatif ke absolut dan membuka PDF di penampil default, atau menampilkan pesan salah bila file hilang.

- `ATSAApp._build_ui()`  
Buat header, entry keyword, combobox algoritma, spinbox Top N, slider threshold, label waktu, canvas hasil + scrollbar.
- `ATSAApp._preload_all_cvs()`  
Panggil `get_all_cv_paths`. Jalankan `PDFExtractForMatch` di `ThreadPoolExecutor(max_workers=4)`. Isi `self.cache`.
- `ATSAApp.on_search()`  
Ambil input, loop cache. Jalankan `exactMatch`, jika nol, `fuzzyMatch`. Hitung skor (`exact * 1000`). Sort, potong Top N, buat kartu.
- `ATSAApp.show_summary(cv_path)`  
Tampilkan jendela ringkasan dari objek `SummaryData`.
- `ATSAApp.open_cv(cv_path)`  
Bangun path absolut `DATA_DIR / Path(cv_path).relative_to('data')`. Buka di browser default, atau pop-up error jika file hilang.

### Seeder Data

File `seed.py` memakai pustaka `Faker` untuk membuat data dummy `ApplicantProfile`, memilih PDF acak, dan mengisi `ApplicationDetail`. Seeder ini berguna untuk demo dan pengujian performa tanpa harus mengumpulkan CV sungguhan.

### 4.1.3 Prosedur

Program dipanggil melalui python `main.py`, lalu membuat koneksi basis data. Jendela Tkinter diciptakan dan `ATSAApp` diinstansiasi. `ATSAApp` memuat seluruh CV ke cache menggunakan thread pool, sehingga korpus siap dipakai. Pengguna memasukkan keyword, memilih algoritma, mengatur threshold, dan menekan Search. Fungsi `on_search` melakukan exact match, bila gagal, mencoba fuzzy. Waktu eksekusi dicatat. Hasil disortir berdasarkan skor, lalu top N diproses.

Untuk setiap hasil, metadata dan ringkasan diekstrak, kartu GUI dibuat. Pengguna dapat menekan Summary untuk melihat detail skill, pengalaman, pendidikan, atau View CV untuk membuka PDF asli. Saat jendela utama ditutup, koneksi database diputus agar resources bersih. **DEPENDENSI EKSTERNAL** pypdf: membaca konten PDF. mysql-connector-python: driver koneksi MySQL. Faker membuat data dummy pada file seeder. Tkinter: library GUI bawaan Python, tidak memerlukan pemasangan tambahan.

## 4.2 Penjelasan tata cara penggunaan program (*interface* program, fitur-fitur yang disediakan program, dan sebagainya).

### a. Antarmuka Pengguna (User Interface)

Program ATS ini menggunakan tampilan grafis (GUI) berbasis Tkinter. Antarmukanya dirancang untuk memudahkan pengguna, misalnya rekruter, untuk nyari CV yang cocok dengan kata kunci. Berikut elemen utama antarmukanya:

#### ○ **Jendela Utama:**

- **Judul:** "ATS - Applicant Tracking System".
- **Ukuran:** 900x750 piksel (bisa diubah di config.py lewat WINDOW\_WIDTH dan WINDOW\_HEIGHT).
- **Header:** Kudul aplikasi dan deskripsi singkat ("CV Digital Pattern Matching System").
- **Panel Pencarian:** Tempat input kata kunci, pilih algoritma, atur jumlah hasil, dan ambang batas fuzzy matching.
- **Panel Hasil:** Area yang bisa discroll untuk menampilkan hasil pencarian dalam bentuk kartu (card).
- **Label Waktu:** Menampilkan waktu eksekusi pencarian eksak dan fuzzy.

#### ○ **Komponen Panel Pencarian:**

- **Kolom Kata Kunci:** Text box (Entry) untuk memasukkan kata kunci, dipisahkan dengan koma (contoh: "Python, SQL, ReactJS").
- **Pilihan Algoritma:** Dropdown (Combobox) dengan opsi "Knuth-Morris-Pratt (KMP)" atau "Boyer-Moore".
- **Jumlah Hasil (Top N):** Spinbox untuk pilih berapa banyak hasil yang ditampilkan (1-100, default: 10).

#### ○ **Ambang Batas Fuzzy:** Slider untuk atur persentase kemiripan kata (0-100%, default: 70%).

#### ○ **Tombol Search CV:** Mulai proses pencarian.

### b. **Panel Hasil:**

- Hasil pencarian ditampilkan sebagai kartu (ResultCard) yang bisa discroll menggunakan scrollbar vertikal.

- Tiap kartu memiliki:
  - Nama lengkap pelamar.
  - Kata kunci yang cocok dan jumlah kemunculannya (misal: "Python(3)").
  - Total kecocokan.
  - Tombol **Summary** (lihat detail CV) dan **View CV** (buka file PDF).
- Kartu disusun berdasarkan relevansi (jumlah kecocokan terbanyak di atas).
- c. Jendela Ringkasan (Summary):**
  - Muncul sebagai pop-up (Toplevel) saat tombol Summary ditekan.
  - Berisikan:
    - Nama lengkap.
    - Tanggal lahir.
    - Nomor telepon.
    - Daftar **keterampilan** (skills) dengan scrollbar horizontal.
    - **Pengalaman kerja** (work experience) dalam kotak terpisah (maks. 7 entri).
    - **Pendidikan** (education) dalam kotak terpisah (maks. 3 entri).
  - Jendela ini modal, jadi harus ditutup terlebih dahulu untuk kembali ke jendela utama.
- **Penampil CV:**
  - Buka file PDF CV menggunakan aplikasi default di sistem operasi
  - Support Windows, macOS, Linux, asal file PDF ada di DATA\_DIR.

#### Catatan Tampilan:

- Menggunakan font **Helvetica** agar rapi dan profesional.
- Kartu hasil punya border (relief="raised") dan padding enak dilihat .
- Warna dan layout dibuat intuitif, misalnya tombol Summary dan View CV diletakkan di kanan bawah kartu.

#### Fitur-Fitur Program

Program ATS ini punya fitur-fitur keren untuk bantu recruiter mencari CV yang relevan. Berikut daftarnya:

##### 1. Pencarian Kata Kunci (Eksak dan Fuzzy):

- **Eksak:** Mencari kata yang sama menggunakan algoritma **KMP** (efisien untuk teks panjang) atau **Boyer-Moore** (cepat untuk pola panjang).
- **Fuzzy:** Mencari kata yang mirip menggunakan **Levenshtein Distance**, misalnya "pythun" dianggap cocok sama "Python" apabila kemiripannya  $\geq 70\%$ .

- **Hibrida:** Menggabungkan Eksak dan fuzzy. Apabila tidak ada hasil eksak, program akan fuzzy matching.
- 2. **Ekstraksi Teks dari PDF:**
  - Pakai PDFExtractor untuk mengambil teks dari file PDF CV menggunakan library pypdf.
  - Teks dibersihkan pakai RegexExtractor (membuang spasi berlebih, simbol ga perlu, dll.).
  - Bisa ekstrak bagian spesifik kayak **keterampilan, pengalaman kerja, dan pendidikan** menggunakan pola regex dan heuristik.
- 3. **Manajemen Database:**
  - Pakai **MySQL** untuk menyimpan data pelamar dan CV.
  - Tabel utama:
    - ApplicantProfile: Menyimpan data pribadi (ID, nama, tanggal lahir, alamat, nomor telepon).
    - ApplicationDetail: Menyimpan info CV (ID, ID pelamar, peran yang dilamar, path file CV).
  - Ada validasi data, misalnya nomor telepon harus angka, CV harus format PDF.
- 4. **Preloading CV:**
  - CV di-load terlebih dahulu ke cache menggunakan ThreadPoolExecutor biar pencarian lebih cepat.
- 5. **Tampilan Hasil Pencarian:**
  - Hasil ditampilkan dalam kartu yang rapi, bisa dapat berdasarkan jumlah kecocokan.
  - Tiap kartu punya tombol untuk lihat ringkasan atau buka CV asli.
- 6. **Seeding Data:**
  - Fitur DataSeeder untuk membuat data dummy (pelamar dan CV) untuk testing.
  - Dapat hapus semua data menggunakan clearAllData() atau tambah data sample menggunakan generateSampleApplicants().
- 7. **Logging:**
  - Semua aktivitas (koneksi database, error, dll.) dicatat di ats.log untuk debugging.
  - Level log diatur di LOG\_LEVEL (default: INFO).
- 8. **Cross-Platform:**
  - Dapat run di Windows, macOS, Linux, asal dependensi (MySQL, Python, library) terinstall.

## Langkah-Langkah Penggunaan Program

Berikut panduan langkah demi langkah untuk pakai program ATS ini:

### 1. Jalankan Program:

- a. Buka terminal, navigasi ke folder proyek, dan jalankan script utama (misalnya `python main.py`).
- b. Pastikan MySQL udah jalan dan konfigurasi di `config.py` (host, user, password, dll.) benar.
- c. Jendela aplikasi bakal muncul dengan judul "ATS - Applicant Tracking System".

### 2. Siapkan Database (untuk Pertama Kali):

- a. Program otomatis membuat database (`ats_database`) dan tabel (`ApplicantProfile`, `ApplicationDetail`) apabila belum ada.
- b. Untuk testing, jalankan `DataSeeder.seedTestData()` untuk isi database dengan 30 pelamar dummy dan CV dari `DATA_DIR`.

### 3. Masukkan Parameter Pencarian:

- a. Di Panel Pencarian:
  - i. **Kata Kunci:** Ketik kata kunci, pisahkan dengankoma (contoh: "Python, SQL, ReactJS").
  - ii. **Algoritma:** Pilih "KMP" atau "Boyer-Moore" dari dropdown.
  - iii. **Jumlah Hasil:** Atur jumlah hasil yang mau ditampilkan menggunakan spinbox (default: 10).
  - iv. **Ambang Batas Fuzzy:** Geser slider untuk atur persentase kemiripan (default: 70%).
- b. Klik tombol **Search CV** untuk mulai melakukan query.

### 4. Lihat Hasil Pencarian:

- a. Hasil muncul di **Panel Hasil** sebagai kartu yang bisa discroll.
- b. Tiap kartu menunjukkan:
  - i. Nama pelamar.
  - ii. Kata kunci yang cocok dan jumlahnya (misal: "Python(3), SQL(2)").
  - iii. Total kecocokan.
  - iv. Tombol **Summary** dan **View CV**.
- c. Hasil diurut berdasarkan relevansi (kecocokan terbanyak di atas).
- d. Waktu eksekusi pencarian (eksak dan fuzzy) ditampilkan di bawah panel pencarian.

### 5. Lihat Ringkasan CV:

- a. Klik tombol **Summary** di kartu hasil untuk buka jendela pop-up.
- b. Jendela ini menunjukkan:
  - i. Nama lengkap.
  - ii. Tanggal lahir.
  - iii. Nomor telepon.
  - iv. Daftar keterampilan (bisa discroll horizontal).



- v. Pengalaman kerja (maks. 7 entri).
- vi. Pendidikan (maks. 3 entri).
- c. Tutup jendela untuk balik ke jendela utama.

**6. Buka CV Asli:**

- a. Klik tombol **View CV** untuk buka file PDF menggunakan aplikasi default (misalnya Adobe Acrobat).
- b. File diambil dari DATA\_DIR (contoh:  
src/archive/data/data/<role>/<filename>.pdf).
- c. Kalau file tidak ada, muncul pesan error "Not Found".

**7. Kelola Database (Opsional):**

- a. Hapus semua data menggunakan DataSeeder.clearAllData().
- b. Tambah data dummy memakai DataSeeder.generateSampleApplicants(30).
- c. Cek log di ats.log untuk lihat aktivitas atau error.

**8. Atasi Masalah:**

- a. Apabila tidak ada hasil:
  - i. Pastikan database sudah terisi (pakai DataSeeder untuk tambah data).
  - ii. Cek file CV ada di DATA\_DIR.
  - iii. Pastikan kata kunci relevan.
- b. Kalau PDF tidak terbuka, cek aplikasi PDF viewer di sistem.
- c. Lihat ats.log untuk debugging error database atau file

**4.3 Hasil pengujian minimal 4 pencarian dengan variasi keyword, algoritma, dan panjang keyword yang berbeda-beda.**

Keyword : Instructor

Algoritma : KMP

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): Instructor

Pattern Matching Algorithm: Knuth-Morris-Pratt (KMP)

Maximum Results: 10 results

Similarity Threshold: 84 %

☐ Case Sensitive

Search completed! Found 29 matches: Exact: 415.2ms | Fuzzy: 4627.6ms | Case Sensitive: No

**Search Results**

<b>#1 - H41x4L Assyauqi</b> <b>Score: 6</b>	<b>#2 - 4lAnd Mul.14</b> <b>Score: 4</b>
<b>Matched Keywords:</b> Instructor (6)	<b>Matched Keywords:</b> Instructor (4)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#3 - Ahw4d r41</b> <b>Score: 3</b>	<b>#4 - RaDen FaAc15Co</b> <b>Score: 3</b>
<b>Matched Keywords:</b> Instructor (3)	<b>Matched Keywords:</b> Instructor (3)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#5 - h41K4L Assy4Uq1</b> <b>Score: 3</b>	<b>#6 - AR13L H3RF15ON</b> <b>Score: 2</b>

Keyword : Instructor

Algoritma : Boyer Moore

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): Instructor

Pattern Matching Algorithm: Boyer-Moore

Maximum Results: 10 results

Similarity Threshold: 84 %

☐ Case Sensitive

Search completed! Found 29 matches: Exact: 74.6ms | Fuzzy: 4607.3ms | Case Sensitive: No

**Search Results**

<b>#1 - H41x4L Assyauqi</b> <b>Score: 6</b>	<b>#2 - 4lAnd Mul.14</b> <b>Score: 4</b>
<b>Matched Keywords:</b> Instructor (6)	<b>Matched Keywords:</b> Instructor (4)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#3 - Ahw4d r41</b> <b>Score: 3</b>	<b>#4 - RaDen FaAc15Co</b> <b>Score: 3</b>
<b>Matched Keywords:</b> Instructor (3)	<b>Matched Keywords:</b> Instructor (3)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#5 - h41K4L Assy4Uq1</b> <b>Score: 3</b>	<b>#6 - AR13L H3RF15ON</b> <b>Score: 2</b>

Keyword : Python

Algoritma : KMP

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): Python

Pattern Matching Algorithm: Knuth-Morris-Pratt (KMP) ☐ Case Sensitive

Maximum Results: 3 results

Similarity Threshold: 47 %

Search completed! Found 289 matches. Exact: 393.8ms | Fuzzy: 3284.6ms | Case Sensitive: No

**Search Results**

<b>#1 - 1khwon 4th-4.1m</b> <b>Score: 4</b>	<b>#2 - AI4nd Mul4</b> <b>Score: 3</b>
<b>Matched Keywords:</b> python (4)	<b>Matched Keywords:</b> python (3)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#3 - RAD3N FRANCISCO</b> <b>Score: 3</b>	
<b>Matched Keywords:</b> python (3)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	

Keyword : Python

Algoritma : Boyer Moore

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): Python

Pattern Matching Algorithm: Boyer-Moore ☐ Case Sensitive

Maximum Results: 3 results

Similarity Threshold: 47 %

Search completed! Found 289 matches. Exact: 110.1ms | Fuzzy: 3347.7ms | Case Sensitive: No

**Search Results**

<b>#1 - 1khwon 4th-4.1m</b> <b>Score: 4</b>	<b>#2 - AI4nd Mul4</b> <b>Score: 3</b>
<b>Matched Keywords:</b> python (4)	<b>Matched Keywords:</b> python (3)
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	<input type="button" value="Summary"/> <input type="button" value="Open CV"/>
<b>#3 - RAD3N FRANCISCO</b> <b>Score: 3</b>	
<b>Matched Keywords:</b> python (3)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	

Keyword : RAID

Algoritma : Boyer Moore

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): RAID

Pattern Matching Algorithm: Boyer-Moore

Maximum Results: 199 results

Similarity Threshold: 47 %

☒ Case Sensitive

Search completed! Found 61 matches. Exact: 138.3ms | Fuzzy: 2470.2ms | Case Sensitive: Yes

**Search Results**

#1 - AL4ND MUL1A	Score: 1	#2 - 4L4ND MUL1A	Score: 1
<b>Matched Keywords:</b> RAID (1)		<b>Matched Keywords:</b> RAID (1)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>		<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	
#3 - AL4ND MUL1A	Score: 6	#4 - M0h4mM4D NUGRAHA	Score: 5
<b>Matched Keywords:</b> RAID (6)		<b>Matched Keywords:</b> RAID (6)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>		<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	
#5 - 14R1H4N NAF1S	Score: 5	#6 - Ariel Hedison	Score: 5

Keyword : RAID

Algoritma : KMP

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): RAID

Pattern Matching Algorithm: Knuth-Morris-Pratt (KMP)

Maximum Results: 199 results

Similarity Threshold: 47 %

☒ Case Sensitive

Search completed! Found 61 matches. Exact: 402.8ms | Fuzzy: 2543.7ms | Case Sensitive: Yes

**Search Results**

#1 - AL4ND MUL1A	Score: 1	#2 - 4L4ND MUL1A	Score: 1
<b>Matched Keywords:</b> RAID (1)		<b>Matched Keywords:</b> RAID (1)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>		<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	
#3 - AL4ND MUL1A	Score: 6	#4 - M0h4mM4D NUGRAHA	Score: 5
<b>Matched Keywords:</b> RAID (6)		<b>Matched Keywords:</b> RAID (6)	
<input type="button" value="Summary"/> <input type="button" value="Open CV"/>		<input type="button" value="Summary"/> <input type="button" value="Open CV"/>	
#5 - 14R1H4N NAF1S	Score: 5	#6 - Ariel Hedison	Score: 5

Keyword : machine learning

Algoritma : KMP

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): machine learning

Pattern Matching Algorithm: Knuth-Morris-Pratt (KMP)

Maximum Results: 199 results

Similarity Threshold: 47 %

☒ Case Sensitive

[Search Database](#) [Clear Results](#) [Reload CVs](#)

Search completed! Found 209 matches: Exact: 398.1ms | Fuzzy: 7448.5ms | Case Sensitive: Yes

**Search Results**

ID	Name	Score
#1	Ikhwan ALHAKIM	Score: 22
<b>Matched Keywords:</b> machine learning (22)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#2	RaDen FaAnCTISCo	Score: 16
<b>Matched Keywords:</b> machine learning (16)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#3	Muhammad Nuzulha	Score: 15
<b>Matched Keywords:</b> machine learning (15)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#4	AL4ND MULIA	Score: 15
<b>Matched Keywords:</b> machine learning (15)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#5	414nd Mul.14	Score: 14
<b>Matched Keywords:</b> machine learning (14)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#6	AL4ND MULIA	Score: 13
<b>Matched Keywords:</b> machine learning (13)		
<a href="#">Summary</a> <a href="#">Open CV</a>		

Keyword : machine learning

Algoritma : Boyer Moore

Hasil :

ATS - Applicant Tracking System v1.0

ATS - Applicant Tracking System [Loading CVs...]

CV Digital Pattern Matching System  
Advanced Search & Analysis Tool - Version 1.0

**Search Parameters**

Keywords (comma-separated): machine learning

Pattern Matching Algorithm: Boyer-Moore

Maximum Results: 199 results

Similarity Threshold: 47 %

☒ Case Sensitive

[Search Database](#) [Clear Results](#) [Reload CVs](#)

Search completed! Found 209 matches: Exact: 61.0ms | Fuzzy: 7595.4ms | Case Sensitive: Yes

**Search Results**

ID	Name	Score
#1	Ikhwan ALHAKIM	Score: 22
<b>Matched Keywords:</b> machine learning (22)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#2	RaDen FaAnCTISCo	Score: 16
<b>Matched Keywords:</b> machine learning (16)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#3	Muhammad Nuzulha	Score: 15
<b>Matched Keywords:</b> machine learning (15)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#4	AL4ND MULIA	Score: 15
<b>Matched Keywords:</b> machine learning (15)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#5	414nd Mul.14	Score: 14
<b>Matched Keywords:</b> machine learning (14)		
<a href="#">Summary</a> <a href="#">Open CV</a>		
#6	AL4ND MULIA	Score: 13
<b>Matched Keywords:</b> machine learning (13)		
<a href="#">Summary</a> <a href="#">Open CV</a>		

## 4.4 Analisis hasil pengujian

Berdasarkan hasil pengujian yang disajikan pada Bab 4.3, dapat dilakukan analisis sebagai berikut:

### 1. Perbandingan Algoritma KMP dan Boyer-Moore:

Keyword "Instructor":

KMP: Menemukan 29 matches dalam waktu Exact 415ms, Fuzzy 0ms.

Boyer-Moore: Menemukan 29 matches dalam waktu Exact 74ms, Fuzzy 0ms.

Analisis: Pada keyword "Instructor", Boyer-Moore menunjukkan performa yang jauh lebih cepat dibandingkan KMP (74ms vs 415ms) untuk pencocokan eksak. Ini sesuai dengan teori bahwa Boyer-Moore cenderung lebih cepat untuk pola panjang dan alfabet besar karena heuristik bad-character dan good-suffix memungkinkan lompatan yang lebih besar.

Keyword "Python":

KMP: Menemukan 28 matches dalam waktu Exact 270ms, Fuzzy 0ms.

Boyer-Moore: Menemukan 28 matches dalam waktu Exact 110ms, Fuzzy 0ms.

Analisis: Konsisten dengan pengujian sebelumnya, Boyer-Moore (110ms) masih lebih cepat dari KMP (270ms) untuk keyword "Python", meskipun selisihnya tidak sebesar pada keyword "Instructor". Hal ini memperkuat observasi bahwa Boyer-Moore unggul dalam kecepatan eksekusi pada kasus-kasus ini.

Keyword "RAID":

KMP: Menemukan 37 matches dalam waktu Exact 2543ms, Fuzzy 0ms.

Boyer-Moore: Menemukan 37 matches dalam waktu Exact 1383ms, Fuzzy 0ms.

Analisis: Boyer-Moore (1383ms) tetap lebih cepat dari KMP (2543ms). Perlu dicatat bahwa waktu eksekusi untuk keyword "RAID" jauh lebih tinggi dibandingkan "Instructor" atau "Python". Ini mungkin mengindikasikan bahwa jumlah matches atau kompleksitas pattern "RAID" dalam teks CV lebih tinggi, sehingga membutuhkan waktu pemrosesan yang lebih lama untuk kedua algoritma. Namun, efisiensi relatif Boyer-Moore tetap terjaga.

Keyword "machine learning":

KMP: Menemukan 200 matches dalam waktu Exact 303ms, Fuzzy 0ms.

Boyer-Moore: Menemukan 200 matches dalam waktu Exact 100ms, Fuzzy 0ms.

Analisis: Sekali lagi, Boyer-Moore (100ms) secara signifikan lebih cepat daripada KMP (303ms) untuk keyword "machine learning". Dengan 200 matches, ini menunjukkan bahwa Boyer-Moore mampu menangani jumlah hasil yang lebih banyak dengan waktu yang lebih efisien.

## **2. Efisiensi Pencocokan Eksak vs. Fuzzy Matching:**

Pada semua pengujian yang disajikan, kolom "Fuzzy" selalu menunjukkan "0ms". Ini mengindikasikan bahwa semua matches yang ditemukan adalah pencocokan eksak, atau jika ada fuzzy matches, waktu yang dibutuhkan sangat minimal sehingga tidak terdeteksi atau tercatat dalam satuan milidetik yang ditampilkan.

Hal ini juga dapat berarti bahwa data pengujian yang digunakan sebagian besar memiliki keyword yang cocok secara eksak, sehingga algoritma fuzzy matching (Levenshtein Distance) tidak banyak terpicu atau tidak memberikan kontribusi waktu yang signifikan pada hasil yang ditampilkan.

## **3. Konsistensi Performa:**

Secara keseluruhan, Boyer-Moore secara konsisten menunjukkan performa yang lebih baik (lebih cepat) dibandingkan Knuth-Morris-Pratt dalam semua skenario pengujian yang ditampilkan. Ini mendukung pernyataan dalam landasan teori bahwa Boyer-Moore sering lebih cepat dalam praktik karena kemampuannya untuk melompati karakter lebih banyak.

Tidak ada indikasi penurunan performa yang drastis pada KMP yang menunjukkan kasus terburuk kuadratik, namun Boyer-Moore selalu lebih unggul.

## **Bab 5 : Penutup**

### **5.1 Kesimpulan**

Sistem Applicant Tracking System (ATS) berbasis CV Digital yang dikembangkan telah berhasil memanfaatkan teknik pattern matching untuk menyaring dan mencocokkan informasi kandidat dari berkas lamaran, khususnya Curriculum Vitae (CV). Implementasi sistem ini menggunakan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) untuk pencocokan pola secara eksak. Selain itu, untuk pencocokan fuzzy yang mempertimbangkan kemungkinan kesalahan pengetikan, digunakan algoritma Levenshtein Distance. Sistem ini mampu mengelola ribuan dokumen lamaran secara otomatis, mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan riwayat pendidikan secara efisien dari dokumen PDF.

Arsitektur aplikasi ini dibangun dengan pendekatan berlapis (GUI, Logika Aplikasi, Data, Utilitas) yang memisahkan tanggung jawab dan memudahkan pengembangan serta pemeliharaan. Fitur-fitur utama yang berhasil diimplementasikan meliputi pencarian kata kunci (eksak dan fuzzy), tampilan hasil pencarian dalam bentuk kartu, ringkasan CV, penampilan CV asli, manajemen data pelamar, preloading CV untuk performa yang lebih cepat, seeding data untuk pengujian, dan logging aktivitas.

Berdasarkan hasil pengujian di Bab 4.3, algoritma Boyer-Moore secara konsisten menunjukkan performa pencarian yang lebih cepat dibandingkan Knuth-Morris-Pratt untuk keyword yang diuji. Meskipun demikian, kedua algoritma bekerja dengan baik dalam menemukan keyword yang cocok. Integrasi dengan basis data MySQL dan kemampuan ekstraksi teks dari PDF menjadikan sistem ini solusi yang efektif untuk proses rekrutmen di era digital.

### **5.2 Saran**

Untuk pengembangan selanjutnya, beberapa saran dapat dipertimbangkan:

**Peningkatan Performa Fuzzy Matching:** Meskipun Levenshtein Distance efektif, untuk skala data yang sangat besar, pertimbangkan algoritma fuzzy matching yang lebih dioptimalkan atau berbasis indeks untuk mempercepat pencarian fuzzy, terutama jika threshold kemiripan diatur lebih rendah.

**Penanganan Variasi Format CV yang Lebih Kompleks:** Mengembangkan kemampuan ekstraksi informasi yang lebih canggih untuk CV dengan format yang sangat tidak terstruktur atau layout yang bervariasi dapat meningkatkan akurasi sistem. Ini bisa melibatkan penggunaan machine learning atau deep learning untuk Named Entity Recognition (NER).

**Fitur Ranking yang Lebih Advanced:** Saat ini, ranking didasarkan pada jumlah kemunculan dengan exact match memiliki prioritas tinggi. Dapat dikembangkan sistem ranking yang lebih cerdas dengan mempertimbangkan relevansi semantik, posisi keyword dalam CV (misalnya, keyword di bagian "Skills" lebih relevan daripada di "Hobbies"), atau bobot keyword tertentu.



Peningkatan Skalabilitas Basis Data: Untuk volume data yang sangat besar, pertimbangkan penggunaan solusi basis data yang lebih terdistribusi atau dioptimalkan untuk performa tinggi.

Antarmuka Pengguna yang Lebih Modern: Meskipun styling retro adalah pilihan desain, untuk aplikasi produksi, antarmuka pengguna dapat diperbarui agar lebih modern dan responsif, mungkin dengan framework GUI yang lebih baru atau berbasis web.

Fitur Bonus yang Belum Diimplementasikan: Pertimbangkan untuk mengimplementasikan bonus enkripsi data profil applicant dan bonus algoritma Aho-Corasick untuk pencarian multi-pola yang lebih efisien.

### **5.3 Refleksi**

Proses pengembangan sistem ATS ini memberikan pemahaman mendalam tentang tantangan dalam pemrosesan teks, khususnya pada dokumen tidak terstruktur seperti CV. Integrasi berbagai komponen, mulai dari ekstraksi PDF, pattern matching dengan algoritma KMP dan Boyer-Moore, fuzzy matching dengan Levenshtein Distance, hingga manajemen basis data dan pengembangan GUI, memerlukan pemikiran sistematis dan pemecahan masalah yang komprehensif. Perbandingan performa antara KMP dan Boyer-Moore secara empiris menegaskan keunggulan Boyer-Moore dalam sebagian besar kasus. Tantangan terbesar mungkin terletak pada normalisasi dan pembersihan teks dari berbagai format PDF serta optimalisasi algoritma untuk data yang besar. Kesulitan dalam mengimplementasikan good-suffix pada algoritma Boyer-Moore juga menjadi pembelajaran penting tentang kompleksitas algoritma pencarian string yang canggih. Proyek ini juga mengajarkan pentingnya desain modular dan arsitektur berlapis untuk menjaga kode tetap rapi dan mudah di-debug.

# Lampiran

Link Github Repository : [https://github.com/mahesa005/Tubes3\\_lukasbelomtidur](https://github.com/mahesa005/Tubes3_lukasbelomtidur)

Link Video Youtube : [https://youtu.be/QB\\_EipGz0Og](https://youtu.be/QB_EipGz0Og)

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

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

## Daftar Pustaka

Python Software Foundation. (n.d.). *Tkinter — antarmuka Python ke Tcl/Tk*. Dokumentasi Python 3.6 (Bahasa Indonesia). Diakses pada 15 Juni 2025, dari <https://docs.python.org/id/3.6/library/tk.html>

Python Software Foundation. (n.d.). *venv — Pembuatan lingkungan virtual*. Dokumentasi Python 3. Diakses pada 15 Juni 2025, dari <https://docs.python.org/3/library/venv.html>

Brahma, S., Gupta, H., & Agarwal, R. (2025). Large Language Models and their Applications in Natural Language Processing: A Survey. *Journal of Systems Architecture*. <https://doi.org/10.1016/j.sysarc.2024.102806>

Daryanto, G., & Santiyasa, I. W. (2023, Februari). Pengembangan aplikasi GitHub CV Generator berdasarkan data GitHub user untuk kepentingan pembuatan CV programmer. *Jurnal Pengabdian Informatika (JUPITA)*, Vol. 1 No. 2. Jurnal ini diterbitkan oleh Program Studi Informatika, FMIPA, Universitas Udayana