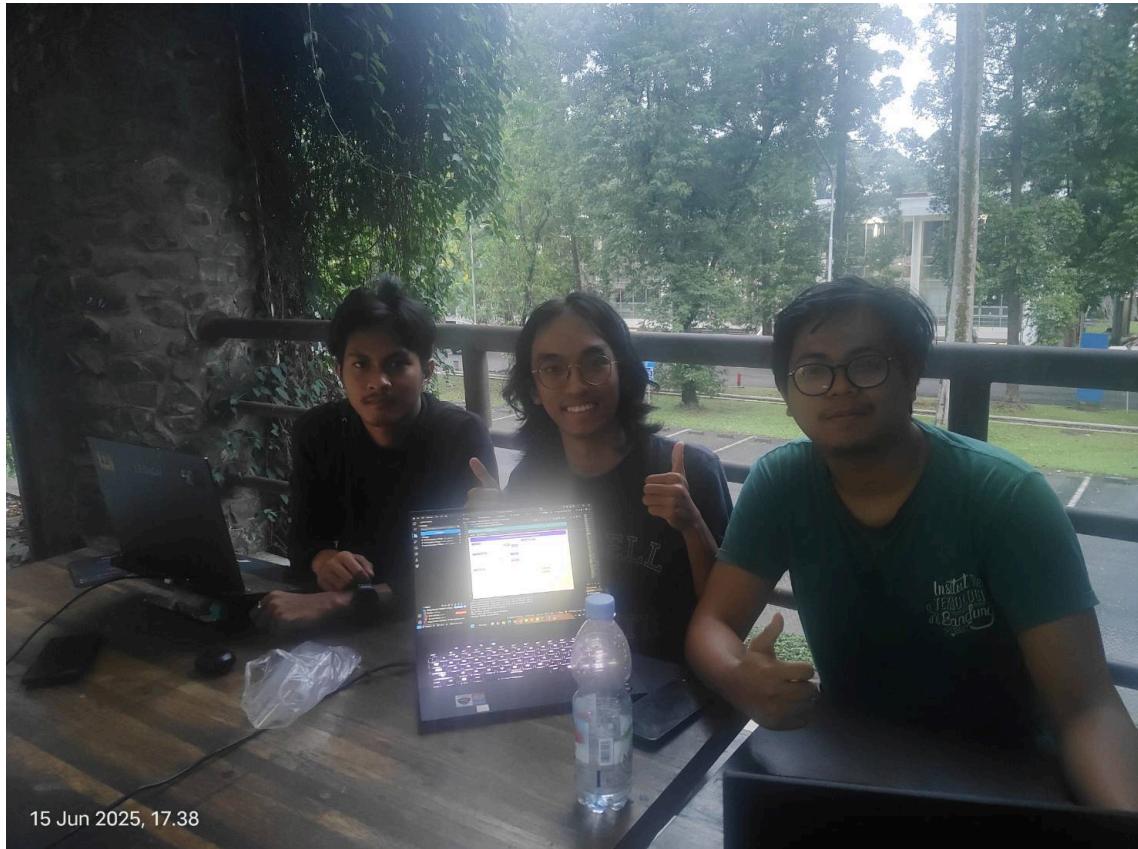


# **Tugas Besar 3 IF2211 - Strategi Algoritma**

## **Semester II Tahun 2024/2025**

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



**Disusun Oleh:**

Muhammad Alfansya	13523005
M. Rayhan Farrukh	13523035
Nadhif Al Rozin	13523076

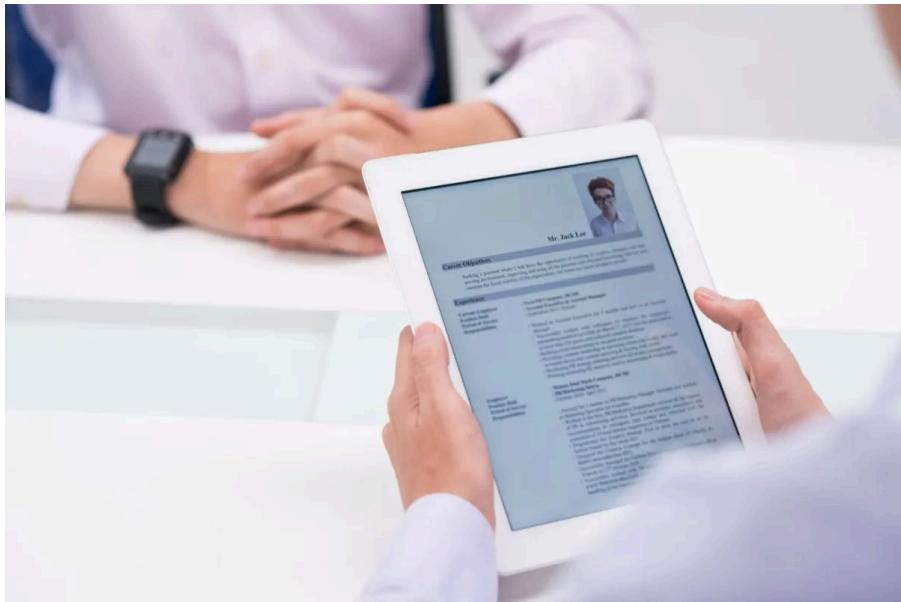
**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2025**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB 1</b>	
<b>DESKRIPSI TUGAS.....</b>	<b>3</b>
<b>Penjelasan Implementasi.....</b>	<b>4</b>
<b>BAB 2</b>	
<b>LANDASAN TEORI.....</b>	<b>6</b>
A. Teori Algoritma.....	6
B. Penjelasan Singkat Aplikasi.....	7
<b>BAB 3</b>	
<b>ANALISIS PEMECAHAN MASALAH.....</b>	<b>8</b>
A. Langkah-langkah pemecahan masalah.....	8
B. Proses pemetaan masalah.....	8
C. Arsitektur Aplikasi.....	11
D. Fitur Fungsional.....	12
<b>BAB 4</b>	
<b>IMPLEMENTASI DAN PENGUJIAN.....</b>	<b>13</b>
A. Spesifikasi Teknis Program.....	13
B. Tata cara penggunaan program.....	47
C. Hasil pengujian.....	50
D. Analisis.....	56
<b>BAB 5</b>	
<b>KESIMPULAN, SARAN DAN REFLEKSI.....</b>	<b>58</b>
A. Kesimpulan.....	58
B. Saran.....	58
C. Refleksi.....	58
<b>BAB 6</b>	
<b>LAMPIRAN.....</b>	<b>59</b>
Tabel Spesifikasi.....	59
DAFTAR PUSTAKA.....	60

## BAB 1

### DESKRIPSI TUGAS



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

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

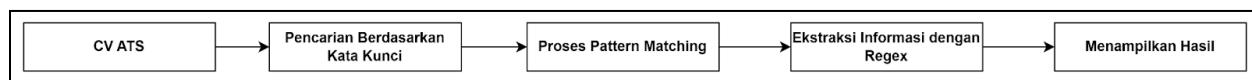
Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan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

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

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

### Penjelasan Implementasi

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



Gambar 2. Skema Implementasi *Applicant Tracking System*

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

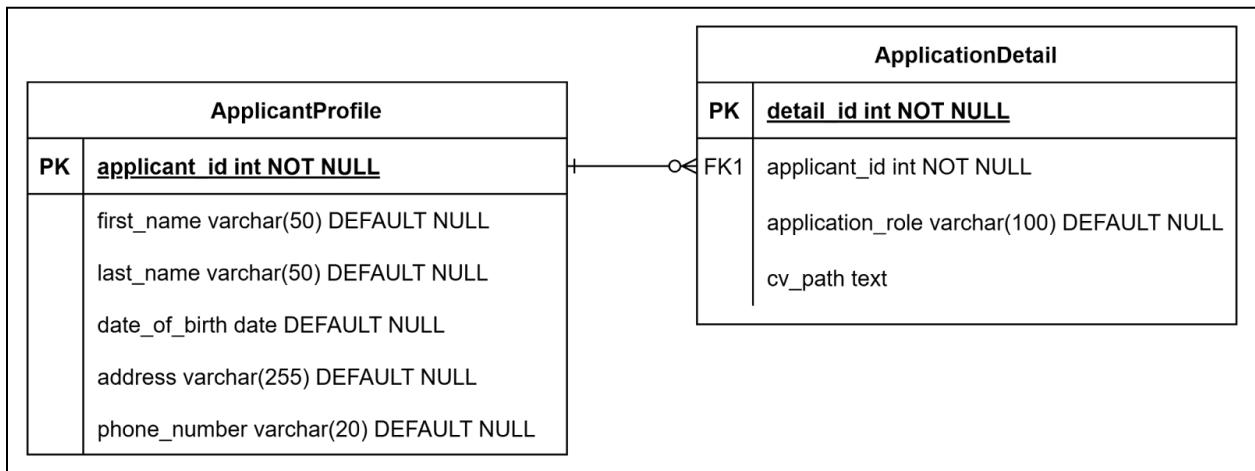
Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

Tabel 1. Hasil ekstraksi teks dari CV ATS

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

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.

## BAB 2

### LANDASAN TEORI

#### A. Teori Algoritma

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

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma *string matching* yang sangat efisien. Tidak seperti metode naif yang menggeser pola satu per satu setelah terjadi *mismatch*, KMP memanfaatkan informasi dari karakter yang sudah cocok sebelumnya untuk melakukan pergeseran yang lebih cerdas dan lebih jauh.

KMP bekerja dengan cara terlebih dahulu melakukan *preprocessing* pada pola untuk membangun sebuah tabel bantu yang sering disebut sebagai *tabel Longest Proper Prefix which is also Suffix* (LPS), atau yang juga disebut *border function table*. Tabel ini menyimpan panjang dari prefiks terpanjang yang juga merupakan sufiks untuk setiap awalan pada pola. Ketika ketidakcocokan terjadi saat membandingkan teks dengan pola, tabel LPS ini akan memberitahu seberapa jauh pola dapat digeser tanpa kehilangan potensi adanya kecocokan. Dengan demikian, algoritma ini tidak perlu mengulang perbandingan pada karakter teks yang sudah terverifikasi cocok, sehingga mencapai kompleksitas waktu yang linier.

##### 2. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah salah satu algoritma *string matching* yang paling populer dan efisien dalam praktiknya. Boyer-Moore memiliki pendekatan yang unik, yaitu melakukan proses pencocokan karakter dari kanan ke kiri pada pola, bukan dari kiri ke kanan seperti kebanyakan algoritma lainnya. Pendekatan ini memungkinkan Boyer-Moore untuk melompati sebagian besar teks dalam sekali pergeseran jika ditemukan *mismatch*, yang membuat Boyer-Moore sangat cepat untuk teks yang panjang.

Efisiensi Boyer-Moore didasarkan pada dua aturan heuristik utama: *bad-character heuristic* dan *good-suffix heuristic*. Ketika terjadi *mismatch*, *bad-character heuristic* akan menggeser pola hingga karakter pada teks yang menyebabkan ketidakcocokan tersebut sejajar dengan kemunculan karakter yang sama paling kanan di dalam pola. Di sisi lain, *good-suffix heuristic* akan menggeser pola berdasarkan sufiks yang sudah berhasil dicocokkan. Boyer-Moore akan memilih pergeseran terjauh dari kedua heuristik ini, sehingga memaksimalkan lompatan dan meminimalkan jumlah perbandingan secara keseluruhan.

##### 3. Algoritma Aho-Corasick

Algoritma Aho-Corasick adalah algoritma *string matching* yang dirancang untuk mencari semua kemunculan dari sekumpulan pola sekaligus secara bersamaan di dalam sebuah teks, berbeda dengan algoritma seperti KMP atau Boyer-Moore yang hanya dapat mencari satu pola saja pada satu waktu. Hal ini membuat algoritma Aho-Corasick berguna untuk pengaplikasiannya seperti *content filtering*, analisis data bioinformatika, atau pendeteksian intrusi jaringan di mana banyak kata kunci harus dipindai secara bersamaan.

Algoritma ini bekerja dengan membangun struktur data yang menyerupai *finite state machine*, yang secara spesifik diimplementasikan sebagai sebuah *trie*. Pada tahap pembangunan, semua pola yang ingin dicari dimasukkan ke dalam trie ini. Setiap node di dalam trie merepresentasikan sebuah prefiks dari satu atau lebih pola. Ketika sebuah pola selesai

dimasukkan, node terakhirnya akan ditandai sebagai *output node* untuk menandakan bahwa sebuah kecocokan penuh telah ditemukan jika alur pencarian mencapai titik tersebut.

Pada tahap pencarian, teks masukan diproses hanya dalam satu kali lintasan (*single pass*). Untuk setiap karakter, algoritma bergerak dari satu *state* (*node*) ke *state* berikutnya di dalam automaton trie. Hal yang membuat Aho-Corasick sangat efisien adalah karena penggunaan *failure links*. Jika pada *state* saat ini tidak ada transisi untuk karakter berikutnya dari teks, algoritma tidak akan kembali ke awal, namun akan mengikuti *failure link*, yang merujuk ke *state* lain. *State* yang ditunjuk oleh *failure links* adalah *node* yang merepresentasikan sufiks terpanjang dari string yang baru saja dicocokkan, yang juga merupakan prefiks dari pola lain di dalam kamus. Mekanisme ini memastikan tidak ada karakter teks yang perlu diperiksa ulang dan membuat pencocokan sangat efisien.

## B. Penjelasan Singkat Aplikasi

Aplikasi ini adalah sebuah sistem ATS yang menggunakan CV Digital berupa file PDF, yang memanfaatkan teknik Pattern Matching untuk mencari hal-hal relevan pada CV. Untuk mencari hal relevan tersebut, aplikasi menggunakan algoritma *pattern matching*, seperti Boyer-Moore, Knuth-Morris-Pratt dan *Aho-Corasick* untuk menganalisis dan mencocokkan pola dalam dokumen CV.

Pengguna dapat memasukkan input daftar kata kunci yang ingin dicari kemunculannya pada CV. Kemudian dari pencarian tersebut, akan ditampilkan hasil pencarian berupa CV *applicant* relevan yang telah diurutkan berdasarkan hasil kecocokan terbanyak. Selain itu, juga ditampilkan lama waktu pencarian. Berdasarkan hasil yang ditampilkan, pengguna dapat melihat *summary* atau ringkasan serta dokumen CV asli dari *applicant* yang dipilih dari hasil pencarian tersebut.

## BAB 3

### ANALISIS PEMECAHAN MASALAH

#### A. Langkah-langkah pemecahan masalah

1. Mengekstrak data *applicant* yang terdapat pada CV
2. Hasil ekstrak akan dimasukkan ke database bersama dengan seeding yang sudah diberikan disesuaikan dengan indeks, filepath tiap cv disimpan sebagai kolom dalam tabel
3. User memasukkan keyword yang nantinya akan dipecah per kata menjadi suatu pola berdasarkan spasi dan “,”.
4. Gunakan salah satu algoritma yang ada untuk mencari kemunculan pola-pola yang dimasukkan pada seluruh teks CV yang telah diekstrak.
5. Jika salah satu pola tidak ditemukan sama sekali kemunculan, coba mencari lagi menggunakan Levenshtein Distance.
6. Hasil akan diurutkan berdasarkan kecocokan dan ditampilkan ke user

#### B. Proses pemetaan masalah

##### 1. KMP

- KMP Border Function

Fungsi pinggiran pada KMP yang dibuat sebelum pencocokan terjadi adalah dengan mencari Suffix terpanjang dari pola dan Prefix terpanjangnya untuk menghindari pengecekan yang tidak perlu

Contoh:

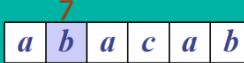
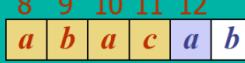
Pattern: ababababca

j	0	1	2	3	4	5	6	7	8	9
P[j]	a	b	a	b	a	b	a	b	c	a
k	0	1	2	3	4	5	6	7	8	
b[k]	0	0	1	2	3	4	5	6	0	

- Pencocokan

Pencocokan dilakukan sama halnya dengan bruteforce, dilakukan lewat kiri namun pergeseran dilakukan jika jumlah karakter yang cocok adalah k maka pencocokan akan dimulai dari index selanjutnya + b[k]

Contoh:

T:	
P:	
	
	
	
<i>j</i>	0 1 2 3 4 5
<i>P[j]</i>	a b a c a b
<i>k</i>	0 0 2 3 4
<i>b(k)</i>	0 0 1 0 1

Sumber:

[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)

## 2. BM

Pencocokan *string* menggunakan Boyer-Moore dilakukan menggunakan *bad character heuristic*. *Bad character heuristic* bekerja dengan cara memasangkan karakter yang ditemukan *mismatch* dengan karakter yang ada pada pola sesuai yang ada pada *last occurrence table*.

- *Last occurrence table*

*Last occurrence table* menyimpan posisi kemunculan terakhir untuk setiap karakter pada pola. Dalam implementasi, *Last occurrence table* didefinisikan sebagai sebuah *dictionary* dengan *key* berupa string yaitu karakter pada pola, dan *value* integer yaitu indeks kemunculan terakhir karakter tersebut.

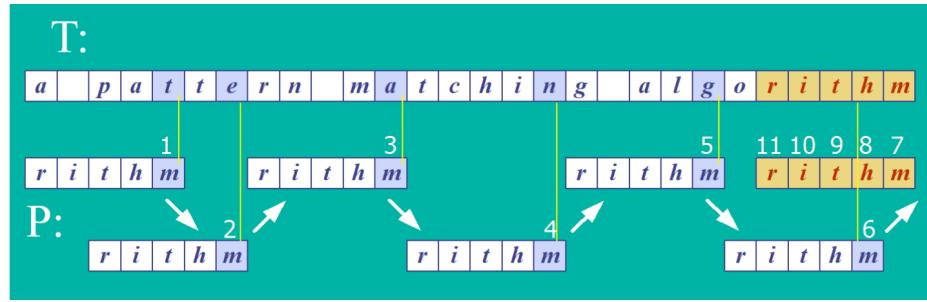
Contoh:

Untuk pola “ABCCGGBN”, maka *last occurrence table*-nya adalah sebagai berikut.

A	B	C	G	N	lainnya
0	5	3	4	6	-1

- Pencocokan

Pelompatan pola pada teks akan disesuaikan dengan apakah karakter yang mismatch pada teks terdapat pada tabel atau tidak, jika ada pada tabel, maka indeks *last occurrence* nya juga akan menjadi perhitungan. Contoh berjalannya algoritma adalah sebagai berikut.



Sumber:

[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)

Algoritma ini akan dijalankan berkali-kali sesuai jumlah *keyword* yang ingin dicari pengguna, yang kemudian akan dilakukan untuk setiap file yang ada pada data sebagai teks yang ingin dicocokkan.

### 3. Aho-Corasick

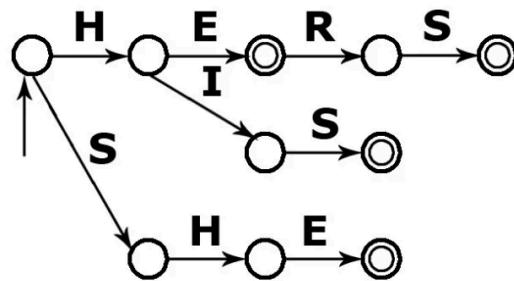
Pada Aho-Corasick, algoritma hanya akan dijalankan sekali untuk setiap teks yang ada. Ini karena Aho-Corasick mampu mencari kemunculan dari beberapa pola sekaligus dalam sekali berjalananya algoritma. Berikut pemetaan permasalahan untuk Aho-Corasick.

- Trie (Goto)

Trie berfungsi sebagai “kamus” terstruktur untuk semua pola yang dicari. Dalam pemetaan ini, seluruh daftar pola yang diberikan oleh pengguna diubah menjadi satu struktur data pohon. Setiap pola menjadi sebuah jalur unik dari *root node* ke *node-node* daun. *Node-node* di dalam Trie merepresentasikan suatu status, dan transisi dari satu status ke status berikutnya ditentukan oleh karakter pada pola-pola.

Contoh:

Jika ada tiga pola yang ingin dicari, yaitu [HERS, HIS, SHE], maka trie yang dibentuk adalah sebagai berikut.



Sumber: [cskane](#)

- Failure Links

*Failure Links* adalah komponen yang membuat Aho-Corasick sangat efisien. Cara kerjanya, untuk setiap status di dalam Trie, sebuah *failure link* dibuat yang menunjuk ke status lain. Status tujuan ini merepresentasikan prefiks terpanjang dari pola lain yang juga merupakan sufiks dari jalur trie yang berhasil dicocokkan sejauh ini. Ketika terjadi *mismatch*, pencocokan tidak kembali ke awal, namun mengikuti *failure link* ini ke jalur lainnya yang terbaik untuk melanjutkan pencocokan dari posisi saat itu.

- Pencocokan

Pencocokan dimulai dari *root node*, lalu mesin bergerak dari satu status ke status lain sesuai trie. Jika terjadi *mismatch*, mesin akan mengikuti fungsi *failure links*. Pada setiap transisi status, mesin akan memeriksa fungsi *output* yang terkait dengan status tersebut. Jika sebuah status adalah akhir dari satu atau lebih kata kunci, maka sebuah kecocokan ditemukan dan dicatat. Dengan cara ini, teks dapat diproses dalam satu kali penelusuran untuk menemukan semua kemunculan dari semua pola secara bersamaan.

## C. Arsitektur Aplikasi

### 1. Frontend

Dengan menggunakan PyQt Designer untuk membuat gambaran kasar desain UI dan dengan menambahkan vanilla CSS untuk memperindah UI. Disiapkan 3 jenis Window yang akan digunakan, Main, Summary dan CV. Main dapat membuka Summary dan CV. Summary dapat membuka CV dan CV adalah endpoint. Main dikelompokkan menjadi beberapa bagian, Upload Section, Database Section, Result Section, Search Section. Summary dipisahkan menjadi Personal Info Section, Summary, Skill, Experience dan Education. CV memiliki dua tombol zoom, in dan out dan CV ditampilkan dalam pdf dengan pymupdf.

Penggunaan Wrapper untuk memoles bagian yang kemungkinan overflow pada skill, education dan experience pada summary window. Wrapper dibuat dengan menggunakan QStyledItemDelegate. Selain itu, untuk meningkatkan User Experience, akan digunakan toast untuk menggantikan window error atau menambahkan toast success sebagai feedback suatu aksi kepada user, hal ini dikombinasikan dengan penggunaan Border Fade pada border input suatu Text yang umumnya menjadi salah satu tempat utama user berinteraksi dengan Program. Toast dibuat dengan memanfaatkan Animation, Timer, Custom Fade in dan Fade out.

Fungsi fungsi yang digunakan pada frontend untuk menggunakan layanan backend disimpan pada [interface.py](#) dengan data structure yang dibuat custom untuk tiap fungsi (Summary data untuk mengambil data Summary suatu CV, Search Data adalah input yang digunakan oleh Front end dan Result Data adalah output yang diharapkan oleh Front end dari Back end).

## 2. Backend

### D. Fitur Fungsional

#### 1. Upload CV

User dapat mengupload cv yang ada. Hal ini dapat berupa folder atau 1 per 1 file pdf

- Folder

- User memiliki folder ALLCV yang berisi semua cv applicant dengan format nama file ID.pdf
- User menekan checkbox folder dan menekan tombol Browse atau CTRL + O
- Cari folder ALLCV dan pilih
- Tekan Upload

- File

- User memiliki file cv bernama tesCV.pdf yang dimiliki oleh applicant dengan id 123123
- User memastikan checkbox folder tidak ditekan, lalu menekan Browse atau CTRL + O
- Pilih file tesCV.pdf
- Masukkan id 123123 ke input yang tertera
- Tekan Upload

#### 2. Pencarian Pola

User dapat memasukkan keyword yang nanti akan dipecah menjadi pola pola untuk dicari ke CV yang ada pada database. Algoritma string matching bisa dipilih oleh user dan jumlah result juga bisa dibatasi

- Masukkan keyword
- Pilih algoritma
- Tentukan jumlah result yang diinginkan
- Tekan clear untuk menghapus

#### 3. Membaca Summary

User dapat membaca summary dari cv yang dipilih pada result. Summary berisi informasi personal, summary, skill, experience dan education

- Pilih cv dari hasil search
- Tekan tombol View Summary

#### 4. Membaca Full CV

User dapat membaca CV full dari window summary atau dari window utama

- Pilih CV hasil search
- Tekan tombol View Full CV
- Jika dari summary, Tekan tombol View Full CV

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN

#### A. Spesifikasi Teknis Program

##### a. Struktur data

```
class SummaryData():
    def __init__(self, nama: str, email: str, phone: str, address: str, skills: List[str], experience: List[str], education: List[str], summary: str):
        self.nama = nama
        self.email = email
        self.phone = phone
        self.address = address
        self.skills = skills
        self.experience = experience
        self.education = education
        self.summary = summary

    def to_string(self) -> str:
        return f"[SUMMARY] Nama: {self.nama}\nEmail: {self.email}\nPhone: {self.phone}\nAddress: {self.address}\nSkills: {', '.join(self.skills)}\nExperience: {', '.join(self.experience)}\nEducation: {', '.join(self.education)}\nSummary: {self.summary}"

class ResultData():
    def __init__(self, id: int, name: str, keywords: dict[str, int]):
        self.id = id
        self.name = name
        self.keywords = keywords

    def to_string(self) -> str:
        keywords_str = ', '.join([f"{key}: {value}" for key, value in self.keywords.items()])
        return f"[RESULT] ID: {self.id}\nName: {self.name}\nKeywords: {keywords_str}"

class searchData():
    def __init__(self, id: int, name: str, text: str):
        self.id = id
        self.name = name
        self.text = text

    def to_string(self) -> str:
        return f"[SEARCH] ID: {self.id}\nName: {self.name}\nText: {self.text}"
```

##### b. UI

```
class CVWindow(QDialog):
    def __init__(self, file_path, parent=None):
        super().__init__(parent)
        self.setWindowTitle("PDF Viewer")
        self.resize(700, 900)
```

```
if not os.path.exists(file_path):
    return

self.doc = fitz.open(file_path)
self.zoom_factor = 1.0

self.scroll = QScrollArea(self)
self.container = QWidget()
self.layout = QVBoxLayout(self.container)

self.labels = []
for page in self.doc:
    pix = page.get_pixmap()
    image = QImage(pix.samples, pix.width, pix.height, pix.stride, QImage.Format_RGB888)
    label = QLabel()
    pixmap = QPixmap.fromImage(image)
    label.setPixmap(pixmap)
    label.originalPixmap = pixmap
    self.layout.addWidget(label)
    self.labels.append(label)

self.scroll.setWidget(self.container)
self.scroll.setWidgetResizable(True)

zoom_in_btn = QPushButton("Zoom In +")
zoom_out_btn = QPushButton("Zoom Out -")

zoom_in_btn.clicked.connect(self.zoom_in)
zoom_out_btn.clicked.connect(self.zoom_out)

btn_layout = QHBoxLayout()
btn_layout.addWidget(zoom_in_btn)
btn_layout.addWidget(zoom_out_btn)

main_layout = QVBoxLayout()
main_layout.addLayout(btn_layout)
main_layout.addWidget(self.scroll)
self.setLayout(main_layout)

def zoom_in(self):
    self.zoom_factor *= 1.25
    self.apply_zoom()

def zoom_out(self):
    self.zoom_factor /= 1.25
    self.apply_zoom()

def apply_zoom(self):
    for label in self.labels:
        originalPixmap = label.originalPixmap
```

```

        size = original_pixmap.size()
        new_size = size * self.zoom_factor
        scaled_pixmap = original_pixmap.scaled(new_size, Qt.KeepAspectRatio, Qt.SmoothTransformation)
        label.setPixmap(scaled_pixmap)

class SummaryWindow(QDialog):
    def __init__(self, parent=None, id=None):
        super().__init__(parent)
        self.ui = Ui_SummaryWindow()
        self.ui.setupUi(self)

        nama, email, phone, address, skills, experience, education, summary = get_summary_data(id)

        self.set_summary_data(summary)
        self.set_name(nama)
        self.set_personal_info(email, phone, address)
        self.set_skills.skills)
        self.set_experience(experience)
        self.set_education(education)
        self.ui.btnExitFullCV.clicked.connect(self.handle_view_cv)

    def handle_view_cv(self):
        file_path = get_file_path(id)

        cv_window = CVWindow(file_path, self)
        cv_window.show()

    def set_summary_data(self, summary_data):
        self.ui.textSummary.setText(summary_data)

    def set_name(self, name):
        self.setWindowTitle(f"Summary for {name}")
        self.ui.lblApplicantName.setText(name)

    def set_personal_info(self, email = None, phone = None, address = None):
        if email:
            self.ui.lblEmail.setText("Email: " + email)
        else:
            self.ui.lblEmail.setText("Email: -")
        if phone:
            self.ui.lblPhone.setText("Phone: " + phone)
        else:
            self.ui.lblPhone.setText("Phone: -")
        if address:
            self.ui.lblAddress.setText("Address: " + address)
        else:
            self.ui.lblAddress.setText("Address: -")

    def set_skills(self, skills):

```

```

print(f"[Skills] : {skills}\n")
self.ui.listSkill.clear()
self.ui.listSkill.setItemDelegate(Wrapper(self.ui.listSkill)) # Pass parent to delegate
self.ui.listSkill.setWordWrap(True)
self.ui.listSkill.setUniformItemSizes(False)
self.ui.listSkill.setSelectionMode(QListWidget.NoSelection)
if skills:
    for skill in skills:
        list_item = QListWidgetItem(skill)
        self.ui.listSkill.addItem(list_item)
else:
    self.ui.listSkill.addItem("No skills listed")

def set_experience(self, experience):
    print(f"[Experience] : {experience}\n") # DEBUG

    self.ui.listExperience.clear()

    self.ui.listExperience.setItemDelegate(Wrapper(self.ui.listExperience)) # Pass parent to delegate

    self.ui.listExperience.setWordWrap(True)
    self.ui.listExperience.setUniformItemSizes(False)
    self.ui.listExperience.setSelectionMode(QListWidget.NoSelection)

    if experience:
        for item_text in experience:
            list_item = QListWidgetItem(item_text)
            self.ui.listExperience.addItem(list_item)
    else:
        self.ui.listExperience.addItem("No work experience listed")

def set_education(self, education):
    print(f"[Education] : {education}\n")
    self.ui.listEducation.clear()
    self.ui.listEducation.setItemDelegate(Wrapper(self.ui.listEducation)) # Pass parent to delegate
    self.ui.listEducation.setWordWrap(True)
    self.ui.listEducation.setUniformItemSizes(False)
    self.ui.listEducation.setSelectionMode(QListWidget.NoSelection)
    if education:
        for item_text in education:
            list_item = QListWidgetItem(item_text)
            self.ui.listEducation.addItem(list_item)
    else:
        self.ui.listEducation.addItem("No education history listed")

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

```

```

# Connect actions to menu items
self.ui.actionUploadCV.triggered.connect(self.handle_action_upload_cv)
self.ui.actionExit.triggered.connect(self.close)
self.ui.actionRefreshDatabase.triggered.connect(self.load_database_info)
self.ui.actionClearDatabase.triggered.connect(self.clear_database)

# Connect buttons to handlers
self.ui.btnBrowse.clicked.connect(self.handle_browse_button)
self.ui.btnUpload.clicked.connect(self.handle_upload_button)
self.ui.btnSearch.clicked.connect(self.handle_search_button)
self.ui.btnClear.clicked.connect(self.handle_clear_button)
self.ui.btnViewSummary.clicked.connect(self.handle_view_summary)
self.ui.btnViewCV.clicked.connect(self.handle_view_cv)

# Initialize UI state
self.ui.btnViewSummary.setEnabled(False)
self.ui.btnViewCV.setEnabled(False)

self.uploaded_cvss = []

# Load initial DB info
self.load_database_info()

## <-----DATABASE HANDLING----->
def clear_database(self):
    response = clear_database()
    if not response:
        toast = Toast("Failed to clear database", duration=3000, parent=self)
        toast.show_above(self)
        return
    else:
        toast = Toast("Database cleared successfully", duration=3000, parent=self)
        toast.show_above(self)
    self.uploaded_cvss.clear()
    self.ui.lblTotalCVs.setText("Total CVs: 0")
    self.ui.lblLastUpload.setText("Last Upload: N/A")
    self.ui.listRecentCVs.clear()
    self.load_database_info()

def load_database_info(self):
    response = load_database()
    if not response:
        toast = Toast("Failed to load database info", duration=3000, parent=self)
        toast.show_above(self)
        return
    total_cvss = len(self.uploaded_cvss)
    last_upload = (
        max(self.uploaded_cvss, key=lambda x: x["upload_time"])["upload_time"].strftime("%Y-%m-%d %H:%M:%S")
        if self.uploaded_cvss else "N/A"
    )

```

```

recent_uploads = [
    f'{cv["filename"]} - {cv["upload_time"].strftime("%Y-%m-%d %H:%M")}'
    for cv in sorted(self.uploaded_cvs, key=lambda x: x["upload_time"], reverse=True)[:5]
]

self.ui.lblTotalCVs.setText(f"Total CVs: {total_cvs}")
self.ui.lblLastUpload.setText(f"Last Upload: {last_upload}")

self.ui.listRecentCVs.clear()
if recent_uploads:
    for item in recent_uploads:
        self.ui.listRecentCVs.addItem(item)
else:
    self.ui.listRecentCVs.addItem("No recent uploads")

## <-----UPLOAD HANDLERS----->
def handle_browse_button(self):
    if self.ui.checkFolderMode.isChecked():
        folder_path = QFileDialog.getExistingDirectory(self, "Select Folder", "")
        if folder_path:
            self.ui.lineEditFilePath.setText(folder_path)
        else:
            toast = Toast("No folder selected.", duration=3000, parent=self)
            toast.show_above(self)
    else:
        file_path, _ = QFileDialog.getOpenFileName(self, "Select File", "", "All Files (*)")
        if file_path:
            self.ui.lineEditFilePath.setText(file_path)
        else:
            toast = Toast("No file selected.", duration=3000, parent=self)
            toast.show_above(self)

def get_file_path(self):
    return self.ui.lineEditFilePath.text().strip()

def fade_border(self, widget=None):
    steps = 10
    interval = 200
    for i in range(steps):
        opacity = 1 - i / steps
        red_value = int(255 * opacity)
        style = f"border: 2px solid rgba({red_value}, 0, 0, 150);"
        QTimer.singleShot(i * interval, lambda s=style: widget.setStyleSheet(s))
    QTimer.singleShot(steps * interval, lambda: widget.setStyleSheet(""))

def get_id_applicant(self):
    text = self.ui.inputIDApplicants.text().strip()
    if not text.isdigit():
        self.fade_border(self.ui.inputIDApplicants)
        toast = Toast("Please enter a valid ID applicant", duration=3000, parent=self)

```

```

        toast.show_above(self)
        return None
    else:
        self.ui.inputIDApplicants.setStyleSheet("")
        return text

def handle_upload_button(self):
    file_path = os.path.basename(self.get_file_path())
    if self.ui.checkFolderMode.isChecked():
        self.handle_upload_folder()
        return
    if not file_path:
        toast = Toast("Please select a file to upload", duration=3000, parent=self)
        toast.show_above(self)
        self.fade_border(self.ui.lineEditFilePath)
        return

    try:
        now = datetime.now()
        filename = file_path.split("/")[-1]
        id_applicant = self.get_id_applicant()

        # TODO: Validate id_applicant before proceeding
        if not id_applicant:
            return

        if not filename.lower().endswith('.pdf'):
            toast = Toast("Only PDF files are allowed", duration=3000, parent=self)
            toast.show_above(self)
            self.fade_border(self.ui.lineEditFilePath)
            self.ui.lineEditFilePath.clear()
            return

        response = add_file(self.get_file_path(), id_applicant)

        if response:
            toast = Toast("File uploaded successfully", duration=3000, parent=self)
            toast.show_above(self)
        else:
            toast = Toast("Failed to upload file", duration=3000, parent=self)
            toast.show_above(self)
            return

        self.uploaded_cvs.append({
            "filename": filename,
            "upload_time": now
        })

        self.ui.lineEditFilePath.clear()
        self.ui.inputIDApplicants.clear()
        print(f"[UPLOAD] File {filename} uploaded successfully at {now}.") # DEBUG

```

```

        self.load_database_info()

    except Exception as e:
        toast = Toast("Failed to upload file:\n" + str(e), duration=3000, parent=self)
        toast.show_above(self)

    def handle_upload_folder(self):
        folder_path = self.get_file_path()
        if not folder_path:
            toast = Toast("Please select a folder to upload", duration=3000, parent=self)
            toast.show_above(self)
            self.fade_border(self.ui.lineEditFilePath)
            return

        try:
            folderName = os.path.basename(folder_path)
            response = add_folder(folder_path, self.uploaded_csvs)

            if response:
                toast = Toast("Folder uploaded successfully", duration=3000, parent=self)
                toast.show_above(self)
            else:
                toast = Toast("Failed to upload file", duration=3000, parent=self)
                toast.show_above(self)
                return
            self.ui.lineEditFilePath.clear()
            print(f"[UPLOAD] Folder {folderName} uploaded successfully.")
        except Exception as e:
            toast = Toast("Failed to upload folder:\n" + str(e), duration=3000, parent=self)
            toast.show_above(self)

    def handle_action_upload_csvs(self):
        self.handle_browse_button()
        if self.get_file_path():
            self.handle_upload_button()

## <-----SEARCH HANDLERS----->
def get_selected_algorithm(self):
    if self.ui.radioKMP.isChecked():
        return "KMP"
    elif self.ui.radioBoyerMoore.isChecked():
        return "BM"
    elif self.ui.radioAhoCorasick.isChecked():
        return "Aho-Corasick"
    return None

def get_result_limit(self):
    return self.ui.spinResultLimit.value()

def get_keywords(self):
    raw_text = self.ui.inputKeywords.text()

```

```

words = re.split(r'[, \s]+', raw_text.strip())
keywords = [word for word in words if word]
print(f"[KEYWORDS] Parsed keywords: {keywords}")

if keywords:
    return keywords
else:
    return []

def handle_search_button(self):
    algorithm = self.get_selected_algorithm()
    keywords = self.get_keywords()
    limit = self.get_result_limit()

    # DEBUG
    print(f"[SEARCH] Searching with {algorithm} for keywords: {', '.join(keywords)} (Limit: {limit})")

    if not keywords:
        self.fade_border(self.ui.inputKeywords)
        toast = Toast("Please enter keyword to search", duration=3000, parent=self)
        toast.show_above(self)
        return

    results, exact_time, fuzzy_time = run_search_algorithm(algorithm, keywords, limit)

    self.ui.listResults.clear()
    for res in results:
        display_text = f"{res.name} (ID: {res.id})"
        if res.keywords:
            display_text += " - Keywords: " + ", ".join(f"{k} ({v})" for k, v in res.keywords.items())
        else:
            display_text += " - No keywords found"
        item = QListWidgetItem(display_text)
        item.setData(Qt.UserRole, res)
        self.ui.listResults.addItem(item)

    if exact_time != None:
        self.ui.lblExactMatchTime.setText("⌚ Exact Match : " + str(exact_time) + " ms")
    else:
        self.ui.lblExactMatchTime.setText("⌚ Exact Match : -ms")

    if fuzzy_time != None:
        self.ui.lblFuzzyMatchTime.setText("🔍 Fuzzy Match : " + str(fuzzy_time) + " ms")
    else:
        self.ui.lblFuzzyMatchTime.setText("🔍 Fuzzy Match : -ms")

    if len(results) > 0:
        self.ui.lblTotalResults.setText(f"📊 Total Results: {len(results)}")

```

```

        else:
            self.ui.lblTotalResults.setText("TOTAL Results: -")

        self.ui.btnViewSummary.setEnabled(len(results) > 0)
        self.ui.btnCloseCV.setEnabled(len(results) > 0)

    def handle_clear_button(self):
        self.ui.inputKeywords.clear()
        self.ui.listResults.clear()
        self.ui.lblExactMatchTime.setText("EXACT Match : -ms")
        self.ui.lblFuzzyMatchTime.setText("FUZZY Match : -ms")
        self.ui.lblTotalResults.setText("TOTAL Results: -")

        self.ui.btnCloseCV.setEnabled(False)
        self.ui.btnCloseCV.setEnabled(False)

## -----RESULT
HANDLERS----->
    def handle_view_summary(self):
        selected = self.get_selected_result()
        if selected:
            print(f"[VIEW SUMMARY] Selected result: name={selected.name}, id={selected.id}")

            summary_window = SummaryWindow(self, id=selected.id)
            summary_window.show()
        else:
            toast = Toast("Please select a result first.", duration=3000, parent=self)
            toast.show_above(self)

    def handle_view_cv(self):
        selected = self.get_selected_result()
        if selected:
            print(f"[VIEW CV] Selected result: name={selected.name}, id={selected.id}")

            file_path = get_file_path(selected.id)

            cv_window = CVWindow(file_path, self)
            cv_window.show()
        else:
            toast = Toast("Please select a result first.", duration=3000, parent=self)
            toast.show_above(self)

    def get_selected_result(self):
        selected_items = self.ui.listResults.selectedItems()
        if selected_items:
            res = selected_items[0].data(Qt.UserRole)
            return res
        return None

if __name__ == "__main__":
    app = QApplication(sys.argv)

```

```
window = MainWindow()
window.show()
sys.exit(app.exec_())

from PyQt5.QtWidgets import QWidget, QVBoxLayout, QLabel
from PyQt5.QtCore import Qt, QPropertyAnimation, QTimer, QPoint, pyqtSignal
from PyQt5.QtGui import QFont

class ToastManager:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
            cls._instance.active_toasts = []
            cls._instance.toast_spacing = -15 # Space between toasts
        return cls._instance

    def add_toast(self, toast):
        self.active_toasts.append(toast)
        self._position_toast(toast)

    def remove_toast(self, toast):
        if toast in self.active_toasts:
            self.active_toasts.remove(toast)
            self._reposition_all_toasts()

    def _position_toast(self, new_toast):
        if len(self.active_toasts) == 1:
            # First
            new_toast.move(new_toast.target_x, new_toast.base_y)
        else:
            lowest_y = new_toast.base_y
            for toast in self.active_toasts[:-1]:
                if toast.isVisible():
                    toast_bottom = toast.y() + toast.height()
                    if toast_bottom > lowest_y:
                        lowest_y = toast_bottom

            # Position new toast below
            new_y = lowest_y + self.toast_spacing
            new_toast.move(new_toast.target_x, new_y)

    def _reposition_all_toasts(self):
        current_y = None
        for toast in self.active_toasts:
            if toast.isVisible():
                if current_y is None:
                    # First goes to base
                    current_y = toast.base_y
```

```
        toast.move(toast.target_x, current_y)
    else:
        current_y = current_y + toast.height() + self.toast_spacing
        toast.move(toast.target_x, current_y)

class Toast(QWidget):
    closing = pyqtSignal()

    def __init__(self, message, duration=3000, parent=None):
        super().__init__(parent)
        self.setWindowFlags(Qt.FramelessWindowHint | Qt.WindowStaysOnTopHint | Qt.Tool)
        self.setAttribute(Qt.WA_TranslucentBackground)
        self.setAttribute(Qt.WA_DeleteOnClose)

        self.setStyleSheet("""
            QWidget {
                background-color: rgba(255, 200, 0, 220);
                color: black;
                border-radius: 10px;
                font-size: 14px;
                font-weight: bold;
            }
        """)
        layout = QVBoxLayout()
        layout.setContentsMargins(20, 15, 20, 15)

        label = QLabel(message)
        label.setAlignment(Qt.AlignCenter)
        label.setWordWrap(True)
        layout.addWidget(label)

        self.setLayout(layout)

        self.setFixedWidth(250)
        self.adjustSize()

        self.manager = ToastManager()
        self.base_y = 0
        self.target_x = 0

        # Animations
        self.fade_in_animation = QPropertyAnimation(self, b>windowOpacity")
        self.fade_in_animation.setDuration(300)
        self.fade_in_animation.setStartValue(0.0)
        self.fade_in_animation.setEndValue(0.95)

        self.fade_out_animation = QPropertyAnimation(self, b>windowOpacity")
        self.fade_out_animation.setDuration(300)
        self.fade_out_animation.setStartValue(0.95)
```

```
    self.fade_out_animation.setEndValue(0.0)
    self.fade_out_animation.finished.connect(self._force_close)

    # Timer
    self.auto_close_timer = QTimer()
    self.auto_close_timer.setSingleShot(True)
    self.auto_close_timer.timeout.connect(self.start_fade_out)
    self.duration = duration

def show_above(self, parent_widget):
    if not parent_widget:
        return

    parent_pos = parent_widget.mapToGlobal(QPoint(0, 0))
    self.target_x = parent_pos.x() + (parent_widget.width() - self.width()) // 2
    self.base_y = parent_pos.y() + 10

    self.show()
    self.raise_()

    self.manager.add_toast(self)

    self.setWindowOpacity(0.0)
    self.fade_in_animation.start()

    # Start timer
    if self.duration > 0:
        self.auto_close_timer.start(self.duration)

def start_fade_out(self):
    self.auto_close_timer.stop()
    self.fade_out_animation.start()

def _force_close(self):
    self.closing.emit()
    self.close()

def closeEvent(self, event):
    if hasattr(self, 'auto_close_timer'):
        self.auto_close_timer.stop()
    if hasattr(self, 'fade_in_animation'):
        self.fade_in_animation.stop()
    if hasattr(self, 'fade_out_animation'):
        self.fade_out_animation.stop()

    if hasattr(self, 'manager'):
        self.manager.remove_toast(self)

    super().closeEvent(event)
```

```
def mousePressEvent(self, event):
    if event.button() == Qt.LeftButton:
        self.start_fade_out()
    super().mousePressEvent(event)

# TESTING FUNCTIONS
def show_toast(parent_widget, message, duration=3000):
    toast = Toast(message, duration, parent_widget)
    toast.show_above(parent_widget)
    return toast

def test_toasts(parent_widget):
    messages = [
        "First toast message",
        "Second toast - should be below first",
        "Third toast - should be at bottom",
        "Fourth toast - stacking test"
    ]

    for i, msg in enumerate(messages):
        QTimer.singleShot(i * 200, lambda m=msg: show_toast(parent_widget, m, 5000))

if __name__ == "__main__":
    import sys
    from PyQt5.QtWidgets import QApplication

    app = QApplication(sys.argv)

    # Create a test parent widget
    parent_widget = QWidget()
    parent_widget.resize(400, 300)
    parent_widget.setWindowTitle("Toast Test")
    parent_widget.show()

    test_toasts(parent_widget)

    sys.exit(app.exec_())

from PyQt5.QtWidgets import QStyledItemDelegate, QStyleOptionViewItem
from PyQt5.QtCore import QSize, Qt
from PyQt5.QtGui import QTextDocument, QPainter

class Wrapper(QStyledItemDelegate):
    def __init__(self, parent=None):
        super().__init__(parent)
        self._cached_sizes = {}

    def paint(self, painter: QPainter, option: QStyleOptionViewItem, index):
        text = index.data(Qt.DisplayRole)
        if not text:
```

```

        return

    doc = QTextDocument()
    doc.setDefaultFont(option.font)
    doc.setTextWidth(option.rect.width())
    doc.setHtml(text)

    painter.save()
    painter.translate(option.rect.topLeft())
    doc.drawContents(painter)
    painter.restore()

def sizeHint(self, option: QStyleOptionViewItem, index):
    text = index.data(Qt.DisplayRole)
    if not text:
        return QSize(0, 0)

    cache_key = (index.row(), text)
    if cache_key in self._cached_sizes:
        return self._cached_sizes[cache_key]

    doc = QTextDocument()
    doc.setDefaultFont(option.font)
    doc.setTextWidth(option.rect.width())
    doc.setHtml(text)

    height = int(doc.size().height())

    self._cached_sizes[cache_key] = QSize(option.rect.width(), height)
    return self._cached_sizes[cache_key]

def clear_cache(self):
    self._cached_sizes.clear()

```

c. Fungsi dan Prosedur  
 - KMP

```

class KMP:
    def __init__(self, pattern:str = "test"):
        self.pattern = pattern
        self.num_occurrences = 0

    def compute_lps(self) -> list[int]:
        m = len(self.pattern)
        lps = [0] * m
        length = 0
        i = 1

```

```

while i < m:
    if self.pattern[i] == self.pattern[length]:
        length += 1
        lps[i] = length
        i += 1
    else:
        if length != 0:
            length = lps[length - 1]
        else:
            lps[i] = 0
            i += 1
return lps

def count_occurrence(self, text) -> int:
    if not self.pattern or not text:
        return 0

    lps = self.compute_lps()
    n = len(text)
    m = len(self.pattern)
    i = 0 # index text
    j = 0 # index pattern

    self.num_occurrences = 0

    while i < n:
        if self.pattern[j] == text[i]:
            i += 1
            j += 1

        if j == m: # Found
            self.num_occurrences += 1
            j = lps[j - 1]
        elif i < n and self.pattern[j] != text[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1

    return self.num_occurrences

def get_num_occurrences(self) -> int:
    return self.num_occurrences

def search_position(self, text) -> list[int]:
    if not self.pattern or not text:
        return []

    positions = []
    lps = self.compute_lps()

```

```

n = len(text)
m = len(self.pattern)
i = 0 # index text
j = 0 # index pattern

while i < n:
    if self.pattern[j] == text[i]:
        i += 1
        j += 1

    if j == m: # Found
        positions.append(i - j)
        j = lps[j - 1]
    elif i < n and self.pattern[j] != text[i]:
        if j != 0:
            j = lps[j - 1]
        else:
            i += 1

return positions

@staticmethod
def search_multi_pattern(text, patterns:list[str]) -> dict[str, int]:
    results = {}
    for pattern in patterns:
        kmp = KMP(pattern)
        count = kmp.count_occurrence(text)
        results[pattern] = count
    return results

```

## - BM

```

class BM:
    # string pattern
    # int occurrences
    def __init__(self, pattern: str):
        self.pattern = pattern
        self.occ_table = self.init_lot()      # holds Last Occurrence Values for all characters in text
        self.occurrences = 0

    def set_pattern(self, pattern: str):
        self.pattern = pattern

    def get_num_occurrences(self) -> int:
        return self.occurrences

    def init_lot(self) -> dict[str, int]:
        """
        Initializes Last Occurrence Table
        """
        occ_table : dict[str, int] = {}

```

```

idx: int = len(self.pattern)-1
for i in range(idx, -1, -1):
    if self.pattern[i] not in occ_table:
        occ_table[self.pattern[i]] = i
return occ_table

def search_positions(self, text: str) -> list[int]:
    pat_len: int = len(self.pattern)
    text_len: int = len(text)

    if pat_len == 0: return []
    if pat_len > text_len: return []

    result: list[int] = []
    i_window_anchor: int = pat_len - 1

    while i_window_anchor < text_len:
        temp_i: int = i_window_anchor

        i: int = i_window_anchor
        j: int = pat_len - 1

        while j >= 0 and text[i] == self.pattern[j]:
            i -= 1
            j -= 1

        if j < 0:
            result.append(i + 1)
            i_window_anchor = temp_i + 1
        else:
            mismatch: str = text[i]
            lx = self.occ_table.get(mismatch, -1)

            shift = j - lx
            i_window_anchor = temp_i + max(1, shift)

    return result

def count_occurrence(self, text: str) -> int:
    result : list[int] = self.search_positions(text)
    self.occurrences = len(result)
    return self.occurrences

@staticmethod
def search_multi_pattern(text: str, patterns: list[str]) -> dict[str, int]:
    results: dict[str, int] = {}      # stores pattern, occurrence
    if not text or not patterns: return results

    for pattern in patterns:
        bm : object = BM(pattern)
        num_occurrences = bm.count_occurrence(text)

```

```

        results[pattern] = num_occurrences
    return results

```

## - Aho Corasick

```

ALPHABET_SIZE : int = 128 # ascii chars
class AhoCorasick:
    def __init__(self, patterns: list[str]):
        self.patterns: list[str] = patterns

        # bitmasks. Index: state, value's bits: which words end in state.
        self.out: list[int] = []

        # index: state, value: state to transition to on mismatch.
        self.fail: list[int] = []

        # State transitions: trie[state][char code] gives the next state.
        self.trie: list[list[int]] = []

    self.build_trie()
    self.compute_failure_links()

    def add_state(self) -> int:
        """Helper to add a new state to all tables."""
        self.trie.append([-1] * ALPHABET_SIZE)
        self.out.append(0)
        self.fail.append(-1)
        return len(self.trie) - 1

    def build_trie(self) -> None:
        """
        Builds the initial Trie from the list of patterns and sets the initial output values.
        """
        root : int = self.add_state() # add state 0

        for i, pattern in enumerate(self.patterns):
            current_state : int = root
            for char in pattern:
                char_code : int = ord(char)
                if self.trie[current_state][char_code] == -1:
                    new_state : int = self.add_state()
                    self.trie[current_state][char_code] = new_state
                current_state = self.trie[current_state][char_code]

            # Mark the end of the pattern with a bitmask
            self.out[current_state] |= (1 << i)

    def compute_failure_links(self):
        """
        Computes failure links for all states and consolidates outputs using BFS.
        """

```

```

"""
queue : list[int] = []
for char_code in range(ALPHABET_SIZE):
    state : int = self.trie[0][char_code]
    if state != -1:
        self.fail[state] = 0
        queue.append(state)
    else:
        self.trie[0][char_code] = 0

while queue:
    state : int = queue.pop(0)
    for char_code in range(ALPHABET_SIZE):
        next_state : int = self.trie[state][char_code]
        if next_state != -1:
            queue.append(next_state)

            # Find failure link for next state
            failure : int = self.fail[state]
            while self.trie[failure][char_code] == -1:
                failure = self.fail[failure]

            self.fail[next_state] = self.trie[failure][char_code]

            # Consolidate outputs from the failure state
            self.out[next_state] |= self.out[self.fail[next_state]]


@staticmethod
def search_multi_pattern(text: str, patterns: list[str]) -> dict[str, int]:
    if not text or not patterns:
        return {p: 0 for p in patterns}

    ac = AhoCorasick(patterns)

    results: dict[str, int] = defaultdict(int)

    current_state: int = 0
    for i, char in enumerate(text):
        char_code: int = ord(char)
        if char_code >= ALPHABET_SIZE:
            current_state = 0
            continue

        while ac.trie[current_state][char_code] == -1:
            current_state = ac.fail[current_state]

        current_state = ac.trie[current_state][char_code]

        if ac.out[current_state] > 0:
            for j in range(len(ac.patterns)):
                if (ac.out[current_state] & (1 << j)):
```

```
        word: str = ac.patterns[j]
        results[word] += 1

final_results = {p: results[p] for p in patterns}

return final_results
```

### - Levenshtein

```
class Levenshtein:
    def __init__(self):
        pass

    def calculate_distance(self, s1: str, s2: str) -> int:
        len_s1 : int = len(s1)
        len_s2 : int = len(s2)

        if len_s1 == 0: return len_s2
        if len_s2 == 0: return len_s1

        dp : list[list[int]] = [[0] * (len_s2 + 1) for _ in range(len_s1 + 1)] # contains distance for
first i and j characters

        for i in range(len_s1 + 1):
            dp[i][0] = i

        for j in range(len_s2 + 1):
            dp[0][j] = j

        for i in range(1, len_s1 + 1):
            for j in range(1, len_s2 + 1):
                cost = 0 if s1[i-1] == s2[j-1] else 1
                dp[i][j] = min(dp[i-1][j] + 1, dp[i][j-1] + 1, dp[i-1][j-1] + cost)

        return dp[len_s1][len_s2]

    def calculate_similarity_percentage(self, s1: str, s2: str) -> float:
        if (len(s1) == 0 and len(s2) == 0) : return 1

        max_len : int = max(len(s1), len(s2))
        distance : int = self.calculate_distance(s1, s2)
        return 1 - (distance/max_len)

    def are_strings_similar(self, s1: str, s2: str, threshold: float) -> bool:
        return self.calculate_similarity_percentage(s1, s2) >= threshold

    def search_positions(self, text: str, pattern: str, threshold: float) -> list[int]:
        result : list[int] = []
        for i in range (0, len(text) - len(pattern) + 1):
            s1 : str = text[i:i+len(pattern)]
```

```

the same size
        if (self.are_strings_similar(s1, pattern, threshold)): # maybe instead of string window,
process each words?
            result.append(i)
    return result

def count_occurrence(self, text: str, pattern:str, threshold: float = 0.8):
    return len(self.search_positions(text, pattern, threshold))

@staticmethod
def search_multi_pattern(text: str, patterns: list[str]) -> dict[str, int]:
    results : dict[str, int] = {}
    if not text or not patterns: return results

    for pattern in patterns:
        lv : object = Levenshtein()
        num_occurrences = lv.count_occurrence(text, pattern)
        results[pattern] = num_occurrences
    return results

```

d. Enkripsi

Enkripsi dilakukan dengan 3 tahap. Pertama menggunakan *Symmetric Cryptography* melalui *Substitution-Permutation Network* (SPN) seperti yang dipakai di AES, namun hanya versi sederhananya. SPN ini digunakan untuk mengenkripsi data-data pada basis data.

Kemudian, untuk menjaga keamanan kunci yang dipakai pada tahap *Symmetric Key*, kunci SPN yang dipakai akan dienkripsi lagi menggunakan *Asymmetric Cryptography*, yaitu *Elliptic Curve Cryptography*. Kemudian kunci yang sudah dienkripsi akan disimpan pada basis data.

- SPN

```

def pad(data: bytes) -> bytes:
    padding_len = BLOCK_SIZE - (len(data) % BLOCK_SIZE)
    padding = bytes([padding_len] * padding_len)
    return data + padding

def unpad(data: bytes) -> bytes:
    padding_len = data[-1]
    if padding_len > BLOCK_SIZE or not all(p == padding_len for p in data[-padding_len:]):
        raise ValueError("Invalid padding")
    return data[:-padding_len]

def expand_key(key: bytes) -> list[bytes]:
    round_keys = []
    for i in range(ROUNDS + 1):
        start = (i * BLOCK_SIZE) % len(key)
        end = start + BLOCK_SIZE
        if end <= len(key):

```

```

        round_keys.append(key[start:end])
    else:
        round_keys.append((key[start:] + key[:end % len(key)]))
    return round_keys

def encrypt(plaintext: str, key: bytes) -> bytes:
    plaintext = plaintext.encode()
    if len(key) < 16:
        raise ValueError("Key must be at least 16 bytes long.")

    round_keys = expand_key(key)
    padded_plaintext = pad(plaintext)
    ciphertext = b''

    for i in range(0, len(padded_plaintext), BLOCK_SIZE):
        block = padded_plaintext[i:i+BLOCK_SIZE]

        block = bytes(a ^ b for a, b in zip(block, round_keys[0]))
        for r in range(1, ROUNDS + 1):
            block = bytes(S_BOX[b] for b in block)
            block = bytes(block[P_BOX[j]] for j in range(BLOCK_SIZE))
            block = bytes(a ^ b for a, b in zip(block, round_keys[r]))

        ciphertext += block

    return ciphertext

def decrypt(ciphertext: bytes, key: bytes) -> str:
    if len(key) < 16:
        raise ValueError("Key must be at least 16 bytes long.")

    round_keys = expand_key(key)
    plaintext = b''

    for i in range(0, len(ciphertext), BLOCK_SIZE):
        block = ciphertext[i:i+BLOCK_SIZE]

        for r in range(ROUNDS, 0, -1):
            block = bytes(a ^ b for a, b in zip(block, round_keys[r]))
            block = bytes(block[INV_P_BOX[j]] for j in range(BLOCK_SIZE))
            block = bytes(INV_S_BOX[b] for b in block)

        block = bytes(a ^ b for a, b in zip(block, round_keys[0]))
        plaintext += block

    return unpad(plaintext).decode()

```

- ECC

```

Point = Optional[Tuple[int, int]]
class EllipticCurve:
    def __init__(self, a : int=a128, b : int=b128, p : int=p128):
        """
        Elliptic curve:  $y^2 \equiv x^3 + ax + b \pmod{p}$ .
        """
        self.a = a
        self.b = b
        self.p = p

    def is_on_curve(self, x : int, y : int) -> bool:
        return (y**2 - (x**3 + self.a * x + self.b)) % self.p == 0

    def point_addition(P : Point, Q : Point, curve : EllipticCurve) -> Point | None:
        if P is None: return Q
        if Q is None: return P

        x1, y1 = P
        x2, y2 = Q

        if x1 == x2:
            if (y1 != y2): return None
            else: return point_doubling(P, curve)

        m = ((y2 - y1) * pow(x2 - x1, -1, curve.p)) % curve.p
        x3 = (m**2 - x1 - x2) % curve.p
        y3 = (m * (x1 - x3) - y1) % curve.p
        return (x3, y3) # Return new point

    def point_doubling(P : Point, curve : EllipticCurve):
        if P is None: return None

        x, y = P

        m = ((3 * x**2 + curve.a) * pow(2 * y, -1, curve.p)) % curve.p
        x3 = (m**2 - 2 * x) % curve.p
        y3 = (m * (x - x3) - y) % curve.p
        return (x3, y3) # Return new point

    def scalar_multiplication(k : int, P : Point, curve : EllipticCurve) -> Point | None:
        """
        k * P on the elliptic curve using the double-and-add algorithm.
        """
        if k == 0 or P is None: return None

        result = None # Start with the point at infinity
        addend = P

        while k:
            if k & 1: # if bit of k == 1
                result = point_addition(result, addend, curve)
            addend = point_doubling(addend, curve)
            k = k // 2

```

```

        result = point_addition(result, addend, curve)
        addend = point_doubling(addend, curve)
        k >>= 1 # divide by 2

    return result

def generate_keys(curve : EllipticCurve, G : Point=G128) -> Tuple[int, Point]:
    d = random.randint(1, curve.p - 1)
    Q = scalar_multiplication(d, G, curve)
    return d, Q

def gen_keystream(seed: int, length: int) -> bytes:
    keystream = bytearray()

    a = 1664525
    c = 1013904223
    m = 2**32

    current_seed = seed
    for _ in range(length):
        current_seed = (a * current_seed + c) % m
        keystream.append(current_seed & 0xFF)
    return bytes(keystream)

def encrypt(plaintext: bytes, public_key: Point, curve: EllipticCurve, G: Point) -> Tuple[Point, bytes]:
    k = random.randint(1, curve.p - 1)
    C1 = scalar_multiplication(k, G, curve)
    kQ = scalar_multiplication(k, public_key, curve)

    shared_x = kQ[0]
    keystream = gen_keystream(shared_x, len(plaintext))

    ciphertext_bytes = bytes([p_byte ^ k_byte for p_byte, k_byte in zip(plaintext, keystream)])

    return C1, ciphertext_bytes

def decrypt(ciphertext: bytes, R: Point, private_key: int, curve: EllipticCurve) -> bytes:
    kQ = scalar_multiplication(private_key, R, curve)

    shared_x = kQ[0]
    keystream = gen_keystream(shared_x, len(ciphertext))
    plaintext = bytes([c_byte ^ k_byte for c_byte, k_byte in zip(ciphertext, keystream)])
    return plaintext

```

#### e. Basis Data

Untuk implementasi enkripsi, pada basis data ditambahkan satu tabel baru, yaitu tabel EncryptionParameters, yang menyimpan parameter dari enkripsi kunci Simetris.

Parameter yang disimpan adalah  $C1_x$  dan  $C1_y$ , yaitu sebuah titik pada *elliptic curve* yang bertindak sebagai *ephemeral public key*, serta *ciphertext* dari kunci SPN. Selain itu, tabel EncryptionParameters dan ApplicantProfile menyimpan data berupa TINYBLOB untuk menyimpan *raw bytes* hasil enkripsi.

```
def extract_text_from_pdf(file_path: str) -> str:
    """
    Mengekstrak teks dari file PDF menjadi satu string panjang menggunakan PyMuPDF.

    :param file_path: Path ke file PDF
    :return: Teks hasil ekstraksi
    """
    try :
        full_text = []

        with fitz.open(file_path) as doc:
            for page in doc:
                text = page.get_text()
                if text:
                    full_text.append(text)

        return "\n".join(full_text)
    except Exception as e:
        print(f"Error extracting text from {file_path}: {e}")
        return ""

def clean_text(text):
    text = text.strip().replace("\r", "").replace("Â", "").replace("ï¼", "")
    return text

def extract_detailed_info(text):
    text = clean_text(text)
    # Summary
    summary_matches =
list(re.finditer(r'(Summary|Objective)(.*?)(Certifications|Skills|Experience|\$)', text,
flags=re.DOTALL))
    if summary_matches:
```

```
        summary = "\n".join([m.group(2).strip() for m in summary_matches if
m.group(2).strip()])
    else:
        summary = 'No summary available'

    # Skills & Highlight
    skills_matches =
list(re.finditer(r'(Skills|Highlight.?).*(Experience|Education|Projects|Skills|$)',

text, flags=re.DOTALL))
    if skills_matches:
        skills = "\n".join([m.group(2).strip() for m in skills_matches if
m.group(2).strip()])
    else:
        skills = 'No skills available'

    # Experience
    experience_matches = list(re.finditer(r'(Experience|Work
History).*(Education|Skills|$)', text, flags=re.DOTALL))
    if experience_matches:
        experience = "\n".join([m.group(2).strip() for m in experience_matches if
m.group(2).strip()])
    else:
        experience = 'No experience listed'

    # Education
    education_matches = list(re.finditer(r'(Education|Academic
Background).*(Skills|Experience|Projects|$)', text, flags=re.DOTALL))
    if education_matches:
        education = "\n".join([m.group(2).strip() for m in education_matches if
m.group(2).strip()])
    else:
        education = 'No education listed'

    return SummaryData(
        nama=None,
        email=None,
        phone=None,
        address=None,
```

```

skills=[s.strip() for s in skills.split('\n') if s.strip()],
experience=[exp.strip() for exp in experience.split('\n') if exp.strip()],
education=[e.strip() for e in education.split('\n') if e.strip()],
summary=summary
)

def extract_summary_data_from_pdf(file_path: str) -> SummaryData:
    extracted_text = extract_text_from_pdf(file_path)
    if isinstance(extracted_text, list):
        extracted_text = "\n".join(extracted_text)
    summary_data = extract_detailed_info(extracted_text)
    return summary_data

# ===== PDF TO MYSQL =====
dotenv.load_dotenv() # Load environment variables from .env file
DB_HOST = os.getenv("DB_HOST")
DB_USER = os.getenv("DB_USER")
DB_PASSWORD = os.getenv("DB_PASSWORD")
DB_NAME = os.getenv("DB_NAME")

def get_connection():
    try :
        temp_conn = mysql.connector.connect(
            host= DB_HOST,
            user = DB_USER,
            password = DB_PASSWORD,
        )
        temp_cursor = temp_conn.cursor()
        temp_cursor.execute(f"CREATE DATABASE IF NOT EXISTS {DB_NAME}")
        temp_conn.commit()
        temp_cursor.close()
        temp_conn.close()

        return mysql.connector.connect(
            host= DB_HOST,
            user = DB_USER,
            password = DB_PASSWORD,

```

```
        database=DB_NAME
    )
except mysql.connector.Error as err:
    print(f"Error: {err}")
    return None

def reset_tables():
    conn = get_connection()
    cursor = conn.cursor()

    # Hapus data anak (ApplicationDetail) dulu karena FK
    cursor.execute("DELETE FROM EncryptionParameters")
    cursor.execute("DELETE FROM ApplicationDetail")
    cursor.execute("DELETE FROM ApplicantProfile")

    cursor.execute("ALTER TABLE ApplicationDetail AUTO_INCREMENT = 1")
    cursor.execute("ALTER TABLE ApplicantProfile AUTO_INCREMENT = 1")

    conn.commit()
    cursor.close()
    conn.close()
    print("Semua data dalam tabel telah dihapus.")

def create_tables_if_not_exist():
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS ApplicantProfile (
            applicant_id INT AUTO_INCREMENT PRIMARY KEY,
            first_name VARCHAR(50) DEFAULT NULL,
            last_name VARCHAR(50) DEFAULT NULL,
            date_of_birth DATE DEFAULT NULL,
            address VARCHAR(255) DEFAULT NULL,
            phone_number VARCHAR(20) DEFAULT NULL
        )
    ''')
    cursor.execute('''


```

```

CREATE TABLE IF NOT EXISTS ApplicationDetail (
    detail_id INT AUTO_INCREMENT PRIMARY KEY,
    applicant_id INT NOT NULL,
    application_role VARCHAR(100) DEFAULT NULL,
    cv_path TEXT,
    FOREIGN KEY (applicant_id) REFERENCES ApplicantProfile(applicant_id)
)
''')
conn.commit()
cursor.close()
conn.close()
print("Tabel sudah dipastikan ada di database.")

def insert_pdf_to_mysql(file_path: str, application_role: str = None):
    fake = Faker("id_ID")
    conn = get_connection()
    cursor = conn.cursor()

    if file_path.lower().endswith('.pdf'):
        print(f"Memproses file: {file_path} =====")

        spnkey = os.urandom(32)

        first_name = ENC.encrypt_spn(fake.first_name(), spnkey)
        last_name = ENC.encrypt_spn(fake.last_name(), spnkey)
        address = ENC.encrypt_spn(fake.address().replace('\n', ', '),
        spnkey)
        phone_number = ENC.encrypt_spn('628' + ''.join(random.choices('0123456789',
k=10)), spnkey)
        date_of_birth = ENC.encrypt_spn(fake.date_of_birth(minimum_age=18,
maximum_age=60).strftime("%Y-%m-%d"), spnkey)
        email = f"{last_name.lower()}.{first_name}@StimeHehe.com"

        cursor.execute(
"""
        INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address,
phone_number)
        VALUES (%s, %s, %s, %s, %s)
        """,

```

```

        (first_name, last_name, date_of_birth, address, phone_number)
    )

applicant_id = cursor.lastrowid # Dapatkan ID hasil insert barusan
(C1_x, C1_y), SPN_key = ENC.encrypt_ecc(spnkey)
cursor.execute(
    """
    INSERT INTO EncryptionParameters (applicant_id, C1_x, C1_y, SPN_key)
    VALUES (%s, %s, %s, %s)
    """,
    (applicant_id, C1_x.to_bytes(32, "big"), C1_y.to_bytes(32, "big"), SPN_key)
)

cursor.execute(
    """
    INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path)
    VALUES (%s, %s, %s)
    """,
    (applicant_id, application_role, file_path)
)
conn.commit()
cursor.close()
conn.close()

def insert_folder_pdfs_to_mysql(folder_path: str, application_role: str = None):
    fake = Faker("id_ID")
    conn = get_connection()
    cursor = conn.cursor()

    for filename in os.listdir(folder_path):
        if filename.lower().endswith('.pdf'):
            print(f"Memproses file: {filename} =====")
            file_path = os.path.join(folder_path, filename)

            spnkey = os.urandom(32)

            first_name = ENC.encrypt_spn(fake.first_name(), spnkey)
            last_name = ENC.encrypt_spn(fake.last_name(), spnkey)

```

```
address = ENC.encrypt_spn(fake.address().replace('\n', ' ', ' '), spnkey)
phone_number = ENC.encrypt_spn('628' + ''.join(random.choices('0123456789', k=10)), spnkey)
date_of_birth = ENC.encrypt_spn(fake.date_of_birth(minimum_age=18, maximum_age=60).strftime("%Y-%m-%d"), spnkey)
email = f"{last_name.lower()}.{first_name}@StimeHehe.com"

cursor.execute(
    """
        INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number)
        VALUES (%s, %s, %s, %s, %s)
        """,
        (first_name, last_name, date_of_birth, address, phone_number)
    )

applicant_id = cursor.lastrowid # Dapatkan ID hasil insert barusan
(C1_x, C1_y), SPN_key = ENC.encrypt_ecc(spnkey)
cursor.execute(
    """
        INSERT INTO EncryptionParameters (applicant_id, C1_x, C1_y, SPN_key)
        VALUES (%s, %s, %s, %s)
        """,
        (applicant_id, C1_x.to_bytes(32, "big"), C1_y.to_bytes(32, "big"),
SPN_key)
    )

cursor.execute(
    """
        INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path)
        VALUES (%s, %s, %s)
        """,
        (applicant_id, application_role, file_path)
    )

conn.commit()
cursor.close()
conn.close()
```

```
print(f"Selesai memasukkan semua file PDF dari {folder_path} ke database.")

def load_search_data_from_sql() -> list:
    from interface import SearchData
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute('''
        SELECT DISTINCT ap.applicant_id, ap.first_name, ap.last_name, ad.cv_path
        FROM ApplicantProfile ap
        JOIN ApplicationDetail ad ON ap.applicant_id = ad.applicant_id
    ''')
    result = []
    rows = cursor.fetchall()

    for row in rows:
        app_id, firstname, lastname, cv_path = row

        cursor.execute('''
            SELECT enc.C1_x, enc.C1_y, enc.SPN_key
            FROM EncryptionParameters enc WHERE enc.applicant_id = %s
        ''', (app_id,))

        enc_params = cursor.fetchall()
        if enc_params:
            key = ENC.decrypt_key_from_id(enc_params[0])
            firstname = ENC.decrypt_spn(firstname, key)
            lastname = ENC.decrypt_spn(lastname, key)

        try:
            text = extract_text_from_pdf(cv_path)
        except Exception:
            text = ''
        result.append(SearchData(id=app_id, name=f'{firstname} {lastname}', text=text))
    cursor.close()
    conn.close()
    return result
```

```
def get_cv_path_by_id(applicant_id: int) -> str:
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute('''
        SELECT ad.cv_path
        FROM ApplicationDetail ad
        WHERE ad.applicant_id = %s LIMIT 1
    ''', (applicant_id,))
    row = cursor.fetchone()
    cursor.close()
    conn.close()
    return row[0] if row else None

def get_summary_by_id(applicant_id: int):
    cv_path = get_cv_path_by_id(applicant_id)
    if not cv_path:
        return None
    data = extract_summary_data_from_pdf(cv_path)

    conn = get_connection()
    cursor = conn.cursor()
    data.email = None

    cursor.execute('''
        SELECT enc.C1_x, enc.C1_y, enc.SPN_key
        FROM EncryptionParameters enc WHERE enc.applicant_id = %s
    ''', (applicant_id,))

    enc_params = cursor.fetchall()
    key = None
    if enc_params:
        key = ENC.decrypt_key_from_id(enc_params[0])

    cursor.execute('SELECT first_name, last_name FROM ApplicantProfile WHERE
applicant_id = %s', (applicant_id,))
    row = cursor.fetchone()
    data.nama = f"{ENC.decrypt_spn(row[0], key) if row else None}"
    {ENC.decrypt_spn(row[1], key) if row else None}"
```

```
cursor.execute('SELECT phone_number FROM ApplicantProfile WHERE applicant_id = %s',
(applicant_id,))
row = cursor.fetchone()
data.phone = ENC.decrypt_spn(row[0], key) if row else None

cursor.execute('SELECT address FROM ApplicantProfile WHERE applicant_id = %s',
(applicant_id,))
row = cursor.fetchone()
data.address = ENC.decrypt_spn(row[0], key) if row else None

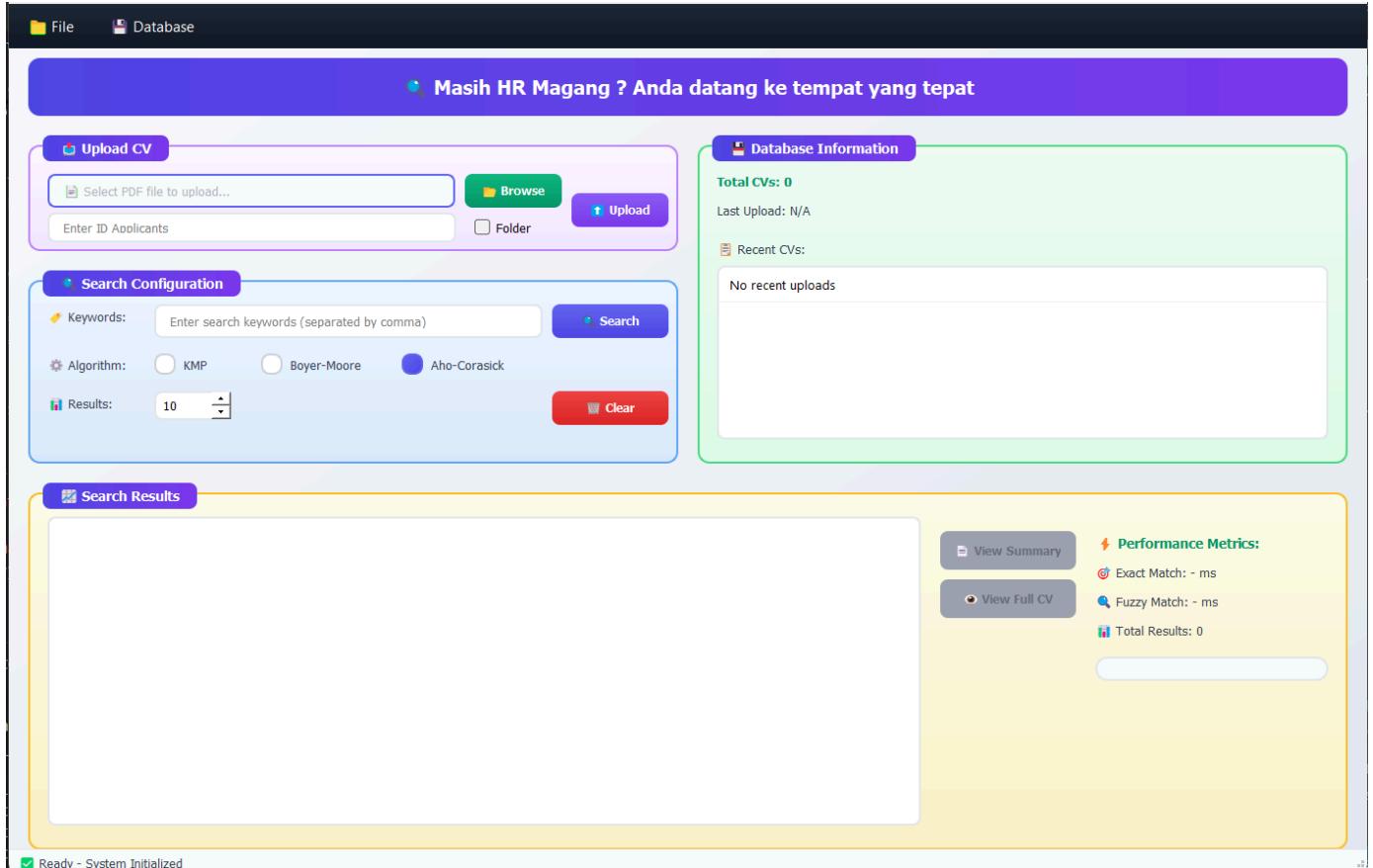
cursor.close()
conn.close()

return data.nama, data.email, data.phone, data.address, data.skills,
data.experience, data.education, data.summary

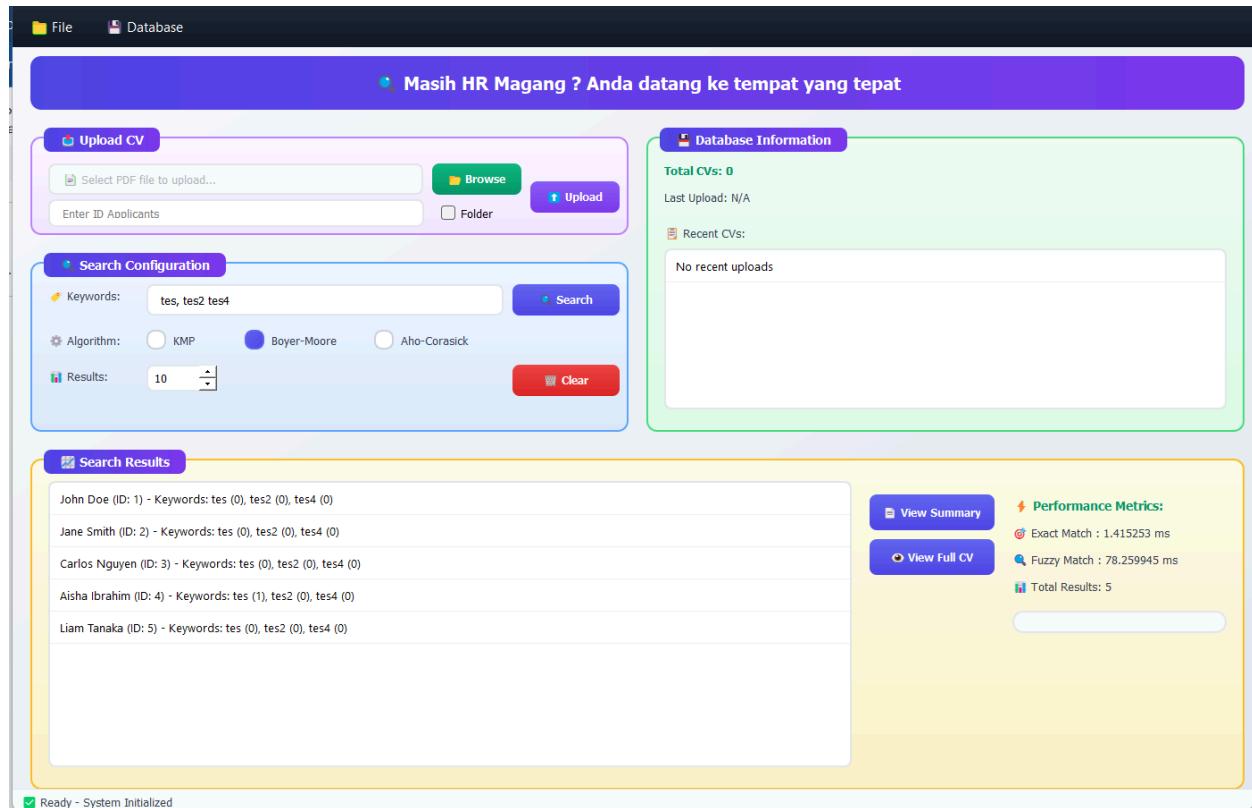
def get_cv_count():
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT COUNT(*) FROM ApplicationDetail")
    count = cursor.fetchone()[0]
    cursor.close()
    conn.close()
    return count
```

## B. Tata cara penggunaan program

- Penjelasan interface



- Hal yang dilakukan pertama adalah mengambil data dari tempat yang kita inginkan (Misalnya folder).
- Tekan checkbox folder lalu tekan tombol Browse.
- Pilih folder yang diinginkan
- Biarkan ID kosong, Tekan Upload
- Akan muncul pesan apakah berhasil/tidak dan dapat dilihat dari Database Information
- Lalu masukkan keyword ke dalam Search, keyword yang dimasukkan akan di cek per kata (dapat dipisahkan dengan spasi dan/atau ,)
- Pilih algoritma yang digunakan dan jumlah hasil yang diinginkan
- Tekan tombol search
- Hasil akan muncul pada Search Result



- Pilih sebuah result untuk melihat summarynya dengan menekan tombol Summary

Email: tes@gmail.com

Phone: 123-456-7890

Address: 123 Main St, City, Country

## Professional Summary

This is a summary of the CV or result being displayed.

Skills

## **Work Experience**

Company A - Data Scientist (2020-2022)  
Company B - Software Engineer (2018-2020)

[View Full CV](#)

 Close

- Atau melihat CV langsung dengan Menekan tombol CV

**Zoom In +**

**Zoom Out -**

**READING TEACHER**

**Summary**  
 I am a highly motivated educator and self-starter with a passionate commitment to learners and their success and growth. I am accountable and thorough with a history of sound decision-making and innovative skills that have helped a plethora of teachers and students succeed.  
 As a professional educator with an extensive background in student success, I am also a certified Life and Success Coach.  
 I look forward to extending my expertise to the teachers and staff at Van Buren Middle School. I believe that student success is inevitable with competent, passionate, "all in" teachers that will stand by their side inspiring productivity and essentially life long learners.

**Highlights**

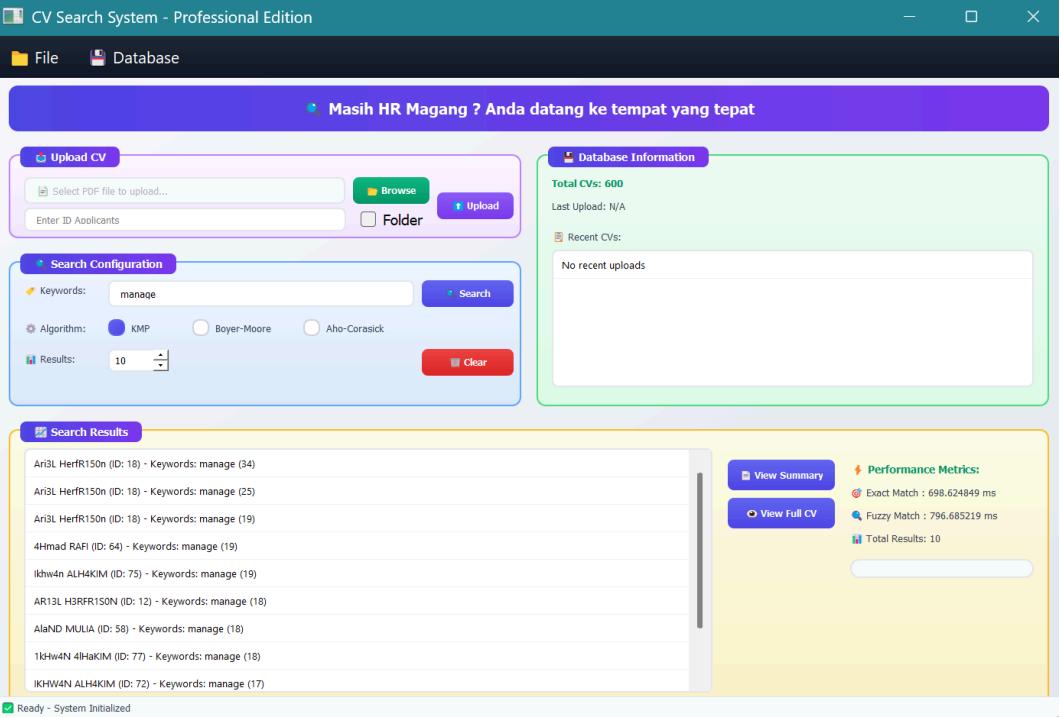
- 97% senior student success rate (2016-2017)
- 15 years of successful teaching experience(tenured)
- Hillsborough Alliance for Black School Educators, HABSE Teacher of the Year 2016-2017
- Teacher of the Year, Greco Middle School 2005-2006
- Rookie Teacher of the Year, Greco Middle School 2002-2003

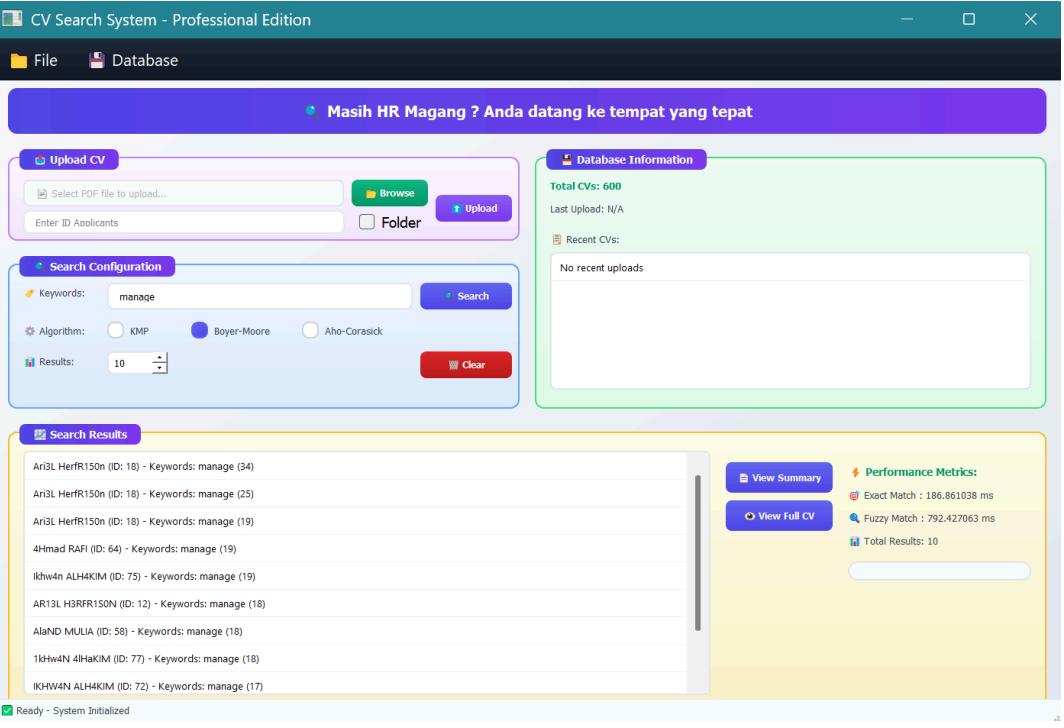
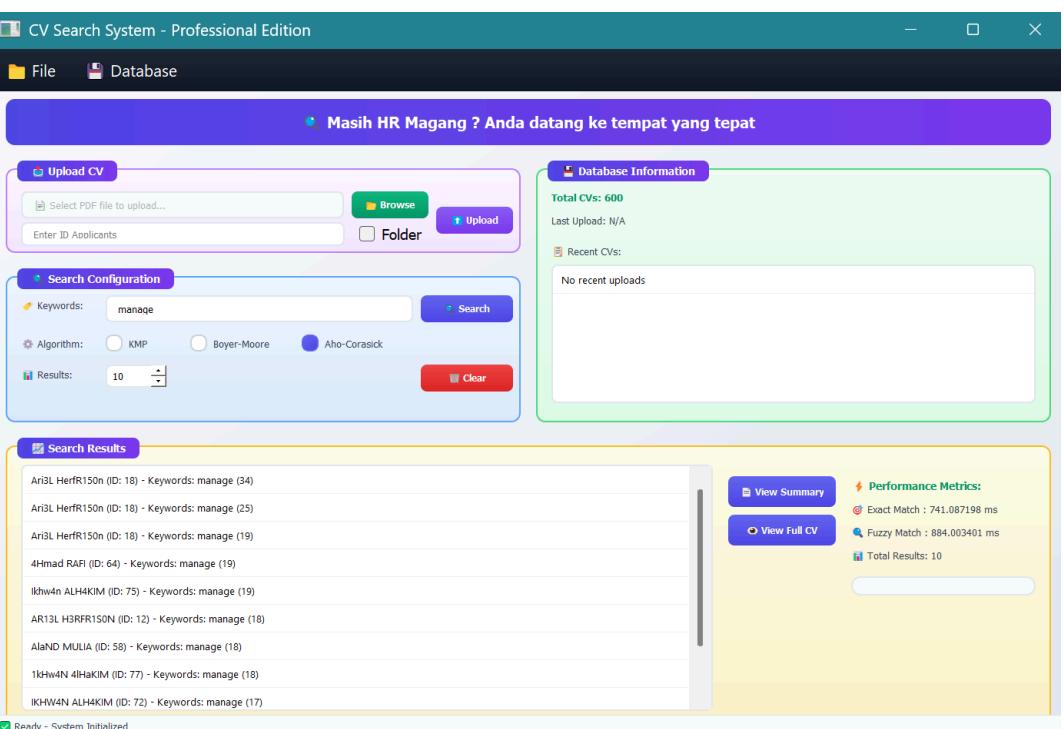
**Experience**  
 Company Name August 2006 to May 2017 Reading Teacher  
 City , State

- Reading Endorsed
- Helped students develop and improve study methods and habits.
- Used a variety of teaching methods such as lectures, discussions and demonstrations.
- Improved 97% reading scores to satisfy graduation requirements.
- Met with parents and guardians to discuss students' progress at least once per semester.
- Established positive relationships with students, parents, colleagues and administrators.

## C. Hasil pengujian

### A. Testcase pattern normal

Algoritma	Tangkapan Layar	Hasil Test
KMP	 <p>Masih HR Magang ? Anda datang ke tempat yang tepat</p> <p><b>Upload CV</b></p> <p>Select PDF file to upload... <input type="button" value="Browse"/> <input type="button" value="Upload"/></p> <p>Enter ID Applicants <input type="checkbox" value="Folder"/></p> <p><b>Search Configuration</b></p> <p>Keywords: manage <input type="button" value="Search"/></p> <p>Algorithm: <input checked="" type="radio"/> KMP <input type="radio"/> Boyer-Moore <input type="radio"/> Aho-Corasick</p> <p>Results: 10 <input type="button" value="Clear"/></p> <p><b>Database Information</b></p> <p>Total CVs: 600 Last Upload: N/A</p> <p>Recent CVs:</p> <p>No recent uploads</p> <p><b>Search Results</b></p> <p>AniL Herr150n (ID: 18) - Keywords: manage (34)      AniL Herr150n (ID: 18) - Keywords: manage (25)      AniL Herr150n (ID: 18) - Keywords: manage (19)      4Hmad RAFI (ID: 64) - Keywords: manage (19)      Ilhw4n ALHAKIM (ID: 75) - Keywords: manage (19)      AR13L H3RFR150N (ID: 12) - Keywords: manage (18)      AlaND MULLA (ID: 58) - Keywords: manage (18)      1Khv4N 4lHaKIM (ID: 77) - Keywords: manage (18)      IKHv4N ALHAKIM (ID: 72) - Keywords: manage (17)</p> <p><input type="checkbox"/> Ready - System Initialized</p> <p><b>Performance Metrics:</b>  <input checked="" type="radio"/> Exact Match : 698.624849 ms  <input type="radio"/> Fuzzy Match : 796.685219 ms  <span style="background-color: yellow;">Total Results: 10</span></p>	waktu Exact Match : 698.624849ms  waktu Fuzzy Match : 796.685219ms

BM		waktu Exact Match : 186.861038ms
Aho Corasick		waktu Exact Time : 741.087198ms

## B. Testcase dua pattern

Algoritma	Tangkapan Layar	Hasil Test
-----------	-----------------	------------

KMP

Masih HR Magang ? Anda datang ke tempat yang tepat

**Upload CV**  
Select PDF file to upload...    
Enter ID Applicants

**Search Configuration**  
Keywords: responsibility development   
Algorithm:  KMP  Boyer-Moore  Aho-Corasick  
Results: 10

**Database Information**  
Total CVs: 600  
Last Upload: N/A  
Recent CVs:  
No recent uploads

**Search Results**  
M0h4mM4D NUGRAHA (ID: 10) - Keywords: responsibility (0), development (34)  
Ari3L Herfr150n (ID: 18) - Keywords: responsibility (0), development (28)  
4HMAD rf1 (ID: 70) - Keywords: responsibility (1), development (14)  
4L4ND MUL1A (ID: 57) - Keywords: responsibility (0), development (13)  
1khw4N 4lHaKIM (ID: 77) - Keywords: responsibility (0), development (12)  
M0h4mM4D NUGRAHA (ID: 10) - Keywords: responsibility (0), development (11)  
IKHWAN Alh4k1M (ID: 79) - Keywords: responsibility (1), development (10)  
MOH4MM4D NUGR4H4 (ID: 2) - Keywords: responsibility (0), development (10)  
Ari3L Herfr150n (ID: 18) - Keywords: responsibility (0), development (10)

**Performance Metrics:**  
Exact Match : 709.644556 ms  
Fuzzy Match : 10214.406013 ms  
Total Results: 10

waktu Exact Match :  
4709.644556ms

Waktu Fuzzy Match:  
10214.406013m  
s

BM

Masih HR Magang ? Anda datang ke tempat yang tepat

**Upload CV**  
Select PDF file to upload...    
Enter ID Applicants

**Search Configuration**  
Keywords: responsibility development   
Algorithm:  KMP  Boyer-Moore  Aho-Corasick  
Results: 10

**Database Information**  
Total CVs: 600  
Last Upload: N/A  
Recent CVs:  
No recent uploads

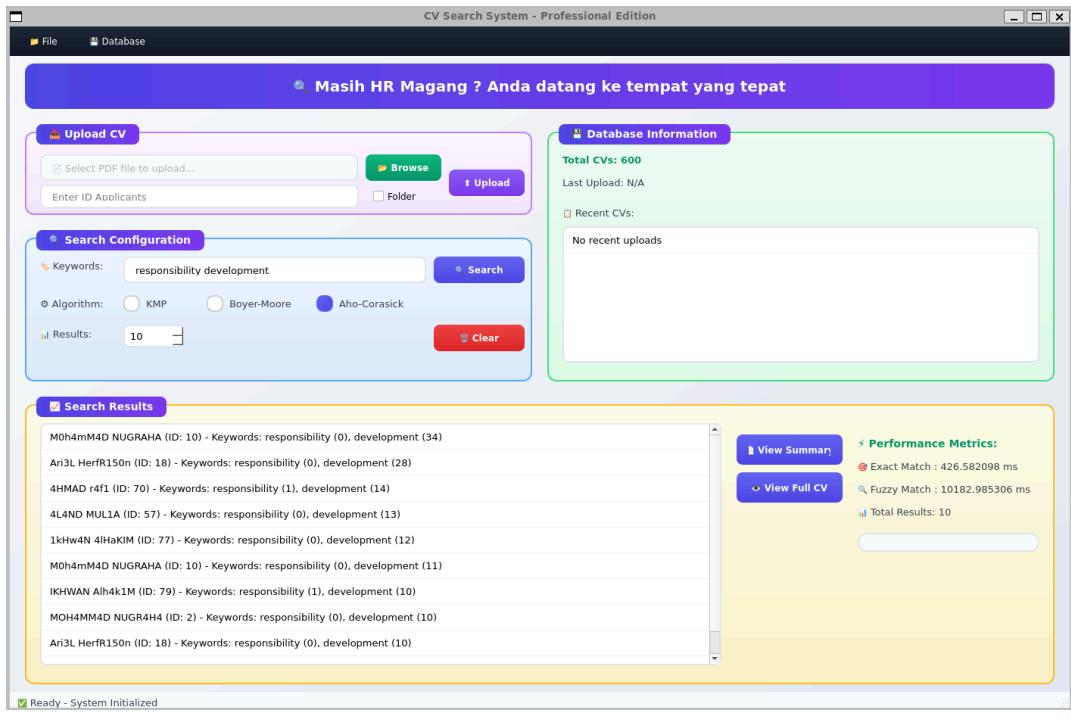
**Search Results**  
M0h4mM4D NUGRAHA (ID: 10) - Keywords: responsibility (0), development (34)  
Ari3L Herfr150n (ID: 18) - Keywords: responsibility (0), development (28)  
4HMAD rf1 (ID: 70) - Keywords: responsibility (1), development (14)  
4L4ND MUL1A (ID: 57) - Keywords: responsibility (0), development (13)  
1khw4N 4lHaKIM (ID: 77) - Keywords: responsibility (0), development (12)  
M0h4mM4D NUGRAHA (ID: 10) - Keywords: responsibility (0), development (11)  
IKHWAN Alh4k1M (ID: 79) - Keywords: responsibility (1), development (10)  
MOH4MM4D NUGR4H4 (ID: 2) - Keywords: responsibility (0), development (10)  
Ari3L Herfr150n (ID: 18) - Keywords: responsibility (0), development (10)

**Performance Metrics:**  
Exact Match : 122.612 ms  
Fuzzy Match : 10215.161562 ms  
Total Results: 10

waktu Exact Match :  
122.612ms

Waktu Fuzzy Match:  
10215.161562m  
s

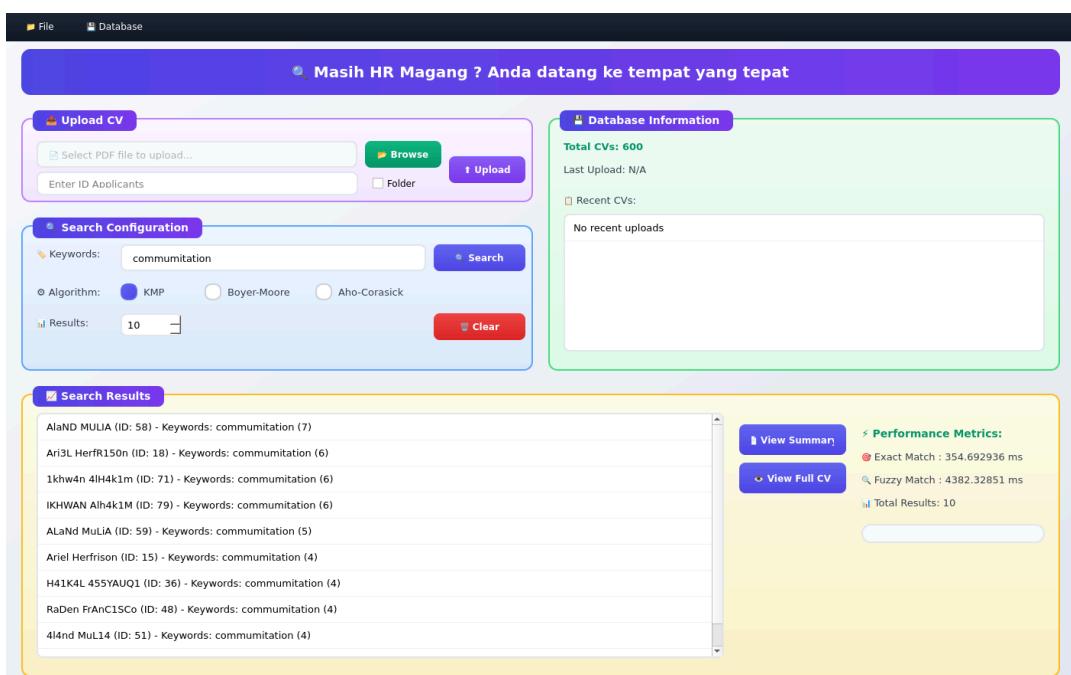
Aho  
Corasick



waktu Exact  
Match :  
426.582098ms

Waktu Fuzzy  
Match:  
10182.985306m  
s

KMP



waktu Exact  
Match :  
354.692936ms

Waktu Fuzzy  
Match:  
4382.32851ms

BM

The screenshot shows the search interface for the Boyer-Moore algorithm. The search term 'communitation' was entered, resulting in 10 matches. The search configuration includes the algorithm set to Boyer-Moore. The database information shows 600 total CVs uploaded.

**Search Results:**

- AlaND MULJA (ID: 58) - Keywords: communitation (7)
- Ari3L HerfR150n (ID: 18) - Keywords: communitation (6)
- 1khwan4lH4k1m (ID: 71) - Keywords: communitation (6)
- IKHWAN Alh4k1M (ID: 79) - Keywords: communitation (6)
- ALaNd MuLiA (ID: 59) - Keywords: communitation (5)
- Ariel Herfrison (ID: 15) - Keywords: communitation (4)
- H41K4L 455YAUQ1 (ID: 36) - Keywords: communitation (4)
- RaDen FrAnC15Co (ID: 48) - Keywords: communitation (4)
- 4l4nd MuL14 (ID: 51) - Keywords: communitation (4)

**Performance Metrics:**

- Exact Match : 61.527252 ms
- Fuzzy Match : 4358.080149 ms
- Total Results: 10

waktu Exact Match :  
61.527252ms

Waktu Fuzzy Match:  
4358.080149ms

Aho Corasick

The screenshot shows the search interface for the Aho-Corasick algorithm. The search term 'communitation' was entered, resulting in 10 matches. The search configuration includes the algorithm set to Aho-Corasick. The database information shows 600 total CVs uploaded.

**Search Results:**

- AlaND MULJA (ID: 58) - Keywords: communitation (7)
- Ari3L HerfR150n (ID: 18) - Keywords: communitation (6)
- 1khwan4lH4k1m (ID: 71) - Keywords: communitation (6)
- IKHWAN Alh4k1M (ID: 79) - Keywords: communitation (6)
- ALaNd MuLiA (ID: 59) - Keywords: communitation (5)
- Ariel Herfrison (ID: 15) - Keywords: communitation (4)
- H41K4L 455YAUQ1 (ID: 36) - Keywords: communitation (4)
- RaDen FrAnC15Co (ID: 48) - Keywords: communitation (4)
- 4l4nd MuL14 (ID: 51) - Keywords: communitation (4)

**Performance Metrics:**

- Exact Match : 390.620232 ms
- Fuzzy Match : 4427.342176 ms
- Total Results: 10

waktu Exact Match :  
390.620232ms

Waktu Fuzzy Match:  
4427.342176ms

#### D. Testcase pattern banyak

KMP

The screenshot shows the HR Magang application interface. At the top, there's a purple header bar with the text "Masih HR Magang ? Anda datang ke tempat yang tepat". Below the header, there are several sections: "Upload CV" (with fields for PDF file upload, ID applicants, and upload button), "Search Configuration" (with keyword input "an staff train unit tax computer work skill now", search button, and algorithm selection for KMP, Boyer-Moore, and Aho-Corasick), and "Database Information" (showing 600 total CVs, last upload N/A, and a list of recent uploads with "No recent uploads"). On the right, there's a "Search Results" section displaying a list of 10 results, each with an ID and keywords. At the bottom right, there are "Performance Metrics" showing Exact Match: 3170.39752ms, Fuzzy Match: 74194.539547ms, and Total Results: 10.

waktu Exact Match :  
3170.39752ms

Waktu Fuzzy Match:  
74194.539547m  
s

BM

The screenshot shows the HR Magang application interface. It has a similar layout to the KMP version, with sections for "Upload CV", "Search Configuration" (using Aho-Corasick algorithm), and "Database Information". The "Search Results" section displays the same 10 results as the KMP version. The "Performance Metrics" at the bottom right show Exact Match: 1368.264914ms, Fuzzy Match: 77736.327648ms, and Total Results: 10.

waktu Exact Match :  
1368.264914ms

Waktu Fuzzy Match:  
77736.327648m  
s

Aho  
Corasick

The screenshot shows a software application interface for searching through a database of CVs. At the top, a purple banner reads "Masih HR Magang ? Anda datang ke tempat yang tepat". Below this, there are two main sections: "Upload CV" and "Search Configuration". The "Search Configuration" section contains fields for "Keywords" (set to "an staff train unit tax computer work skill now"), "Algorithm" (set to "Aho-Corasick" with radio buttons for KMP and Boyer-Moore), and "Results" (set to 10). To the right, a "Database Information" panel shows "Total CVs: 600" and "Last Upload: N/A". Below these are sections for "Recent CVs" (empty) and "Performance Metrics" (Exact Match: 540.009975 ms, Fuzzy Match: 75415.828466 ms). The bottom half of the screen displays a table of search results, each row showing a CV ID and its keywords.

waktu Exact  
Match :  
540.009975ms

Waktu Fuzzy  
Match:  
75415.828466m  
s

## D. Analisis

Dari percobaan A dengan keyword ‘manage’ waktu tercepat didapat dengan algoritma BM(86.861038ms) dan kedua dengan algoritma KMP (698.624849ms) dan Aho Corasick (741.087198ms). Hal ini karena algoritma BM menggunakan *table last occurrence* dan akan melompat sejauh panjang pattern jika huruf yang dicek tidak pernah muncul di teks, KMP akan membandingkan satu persatu meskipun huruf yang dicek tidak akan muncul di keyword

Pada percobaan B dengan dua pattern ‘responsibility development’ yang mendapat waktu tercepat adalah BM (122.612ms) dan kemudian Aho Corasick (426.582098ms) karena, seperti percobaan A, algoritma BM cepat untuk pattern panjang dan algoritma Aho Corasick cepat untuk multiple pattern, algoritma KMP mendapat waktu yang cukup jauh lebih lambat dibanding kedua algoritma lain yaitu 4709.644556ms

Pada percobaan C, dengan keyword typo ‘commummitation’, exact match tercepat dicapai oleh algoritma Boyer Moore (61.527252ms), diikuti oleh KMP (354.692936ms) dan Aho Corasick (390.620232ms). Ini menunjukkan bahwa Boyer Moore sangat unggul untuk mencari satu keyword yang ukurannya cukup panjang. Namun, fokus dari test ini adalah pada *fuzzy matching*, dimana ketiga percobaan menghasilkan waktu yang relatif sama, yaitu 4358 - 4427ms.

Pada percobaan D, dimana keyword yang dites terdiri dari 7 keyword berbeda dengan panjang yang berbeda-beda, terlihat bahwa *bottleneck* terbesar dari aplikasi adalah pada *fuzzy matching*, dengan hasil waktu ada di sekitar 75 detik. Namun untuk *exact match*, pada percobaan ini kita dapat melihat keunggulan dari algoritma Aho Corasick, yaitu *single pass* untuk keyword yang banyak. Oleh karena itu, Aho Corasick berhasil menyelesaikan

pencocokan dalam waktu tercepat (540.009975ms), diikuti oleh Boyer Moore (1368.264914ms) dan KMP dengan waktu *exact match* paling lama (3170.39752ms).

## **BAB 5**

### **KESIMPULAN, SARAN DAN REFLEKSI**

#### **A. Kesimpulan**

Penggunaan algoritma Aho-corasick lebih cepat pada kasus kata kunci yang dimasukkan banyak. Saat menggunakan keyword yang pendek, algoritma KMP lebih cepat daripada Boyer Moore. Sedangkan jika menggunakan keyword yang lebih panjang Algoritma Boyer Moore lebih baik daripada menggunakan KMP. Hasil yang didapat dari aplikasi ini telah sesuai dengan teori, dimana Aho Corasick mampu mencari banyak kata kunci dalam *single pass*. Serta Boyer Moore yang dapat melakukan *skip* ketika ditemukan *mismatch*. Namun, ketika pada suatu dokumen tidak ditemukan kemunculan pada tahap *exact matching*, aplikasi akan melakukan *fuzzy matching*. Karena ada banyak dokumen yang diproses, dengan konten yang beragam, serta penghitungan *Levenshtein Distance* yang tidak secepat algoritma *exact match*, proses *fuzzy matching* ini menjadi *bottleneck* terbesar aplikasi ini.

Dari hasil tersebut, dan melihat penggunaan yang relevan dan realistik dari aplikasi ini, maka dapat disimpulkan algoritma Aho Corasick-lah yang cocok untuk digunakan pada kebanyakan kasus, jika diinginkan kecepatan. Ini karena ketika kita mencari banyak kata kunci, maka lebih besar kemungkinan ditemukan *exact match*, sehingga Aho Corasick yang cepat, serta proses *fuzzy matching* diminimalkan, aplikasi menjadi lebih cepat.

#### **B. Saran**

Aplikasi yang kami buat masih jauh dari sempurna, masih banyak fitur-fitur yang masih bisa dikembangkan lagi seperti tampilan aplikasi, perbaikan algoritma yang digunakan, serta tambahan algoritma-algoritma lain dan fitur yang bisa lebih berguna untuk pengguna umum. Kami juga bisa memperbaiki *user experience* dari aplikasi agar dapat lebih nyaman digunakan oleh pengguna.

#### **C. Refleksi**

Dalam mengembangkan aplikasi ini, kami mendapat pengalaman dan pembelajaran dalam hal pengembangan antarmuka aplikasi dalam *python* serta konsep-konsep algoritma menarik, khususnya pada algoritma Aho Corasick yang memerlukan pemahaman tentang konsep *finite state machine*. Selain itu, kami juga berhasil mengimplementasikan algoritma kriptografi simetris dan asimetris untuk melakukan pengamanan data dengan sistem *hybrid cryptography* sehingga data *applicant* yang disimpan menjadi lebih aman.

## BAB 6

### LAMPIRAN

Repositori GitHub : [https://github.com/Narrr21/Tubes3\\_HR\\_Magang](https://github.com/Narrr21/Tubes3_HR_Magang)

Link Youtube: [https://youtu.be/KJO5aRVa1f8?si=\\_zvF52Mz\\_caCNj3](https://youtu.be/KJO5aRVa1f8?si=_zvF52Mz_caCNj3)

**Tabel Spesifikasi**

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

## DAFTAR PUSTAKA

- [1] R. Munir, "Pencocokan String (String/Pattern Matching)" Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Daring]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf). [Diakses: 12 Juni 2025].
- [2] GeeksforGeeks, "Aho-Corasick Algorithm for Pattern Searching," *GeeksforGeeks*, 13 Juni 2024. [Daring]. Tersedia: <https://www.geeksforgeeks.org/dsa/aho-corasick-algorithm-pattern-searching/>. [Diakses: 12 Juni 2025].